PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-229391
(43)Date of publication of application : 24.08.2001

(51)Int.Cl. G06T 13/00
// H04N 7/24

(21)Application number : 2000-367633
(22)Date of filing : 01.12.2000

(71)Applicant : CANON INC
(72)Inventor : DORRELL ANDREW

(30)Priority

Priority number	Priority date	Priority country
1999 PQ4415	02.12.1999	AU
2000 PQ7724	24.05.2000	AU

(54) METHOD FOR ENCODING ANIMATION INCLUDED IN IMAGE FILE

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a method for encoding an animation included in an image file.
SOLUTION: This method is used to process a multi-layer image file (100) including a plurality of 1st image layers (106-108) and a plurality of 2nd image layers (120-122), to generate an animation sequence, to process an image layer (106, for example) based on its corresponding control block (120, for example) and to provide the images for the animation sequence. In the case where the image layer (106) should be used again in an image sequence, a tag is given to the layer (106) by means of the address of the corresponding control block (120) (i) and also the relative address to one of addresses of control blocks corresponding to other image layers (ii).

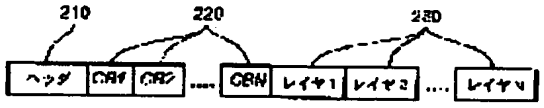


Fig. 2

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－229391
（P2001－229391A）
(43)公開日　平成13年８月24日(2001.8.24)

(51)Int.Cl.'	識別記号	ＦＩ	テーマコード'（参考）
Ｇ０６Ｔ　13/00		Ｇ０６Ｔ　13/00	Ｂ
// Ｈ０４Ｎ　7/24		Ｈ０４Ｎ　7/13	Ｚ

審査請求　未請求　請求項の数19　ＯＬ　外国語出願　（全 93 頁）

(21)出願番号　特願2000－367633(P2000－367633)
(22)出願日　平成12年12月１日(2000.12.1)
(31)優先権主張番号　ＰＱ４４１５
(32)優先日　平成11年12月２日(1999.12.2)
(33)優先権主張国　オーストラリア（ＡＵ）
(31)優先権主張番号　ＰＱ７７２４
(32)優先日　平成12年５月24日(2000.5.24)
(33)優先権主張国　オーストラリア（ＡＵ）

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号
(72)発明者　アンドリュー　ドレル
オーストラリア国　2113　ニュー　サウス　ウェールズ州　ノース　ライド，　トーマス　ホルト　ドライブ　１　キヤノン　インフォメーション　システムズ　リサーチ　オーストラリア　プロプライエタリー　リミテツド内
(74)代理人　100076428
弁理士　大塚　康徳　（外２名）

(54)【発明の名称】　画像ファイル中のアニメーションの符号化方法

(57)【要約】
【課題】　画像ファイルにおけるアニメーションを符号化するための方法を提供すること。
【解決手段】　第１の複数の画像レイヤ（１０６～１０８）、および第２の複数の制御ブロック（１２０～１２２）を含むマルチレイヤ画像ファイル（１００）を処理する方法が開示される。本処理はアニメーションシーケンスを生成する。本方法は、対応する制御ブロック（たとえば１２０）に従った画像レイヤ（たとえば１０６）の処理、それによる前記アニメーションシーケンスのための画像の提供を含む。本方法はさらに、画像レイヤ（１０６）が画像シーケンス中で再度使用されるべきである場合には、再処理のために画像レイヤ（１０６）にタグ付けし、前記タグ付けは、（ｉ）対応する制御ブロック（１２０）のアドレス、および（ｉｉ）別の画像レイヤに対応する制御ブロックのアドレスのうちの１つに向けられる相対アドレスを使用する。

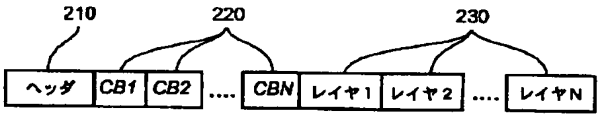


Fig. 2

1

【特許請求の範囲】

【請求項１】　（ｉ）第１の複数の画像レイヤおよび（ｉｉ）第２の複数の制御ブロックを含んでいるマルチレイヤ画像ファイルを処理する方法であって、前記処理はアニメーションシーケンスを生成し、

前記画像ファイル中に複数の制御ブロックを提供するステップであって、各制御ブロックは少なくとも１つの前記画像レイヤに関連しており、各制御ブロックは、前記制御ブロックおよび関連する画像レイヤのいずれか１つがループバックすべきかを示す情報制御フィールドによって特徴付けられ、

各制御ブロックを連続的に実行し、前記情報制御フィールドによって提供される指示に従って、前記実行シーケンス中の前の制御ブロックおよび関連するレイヤへループバックするステップとを含むことを特徴とする方法。

【請求項２】　前記制御ブロックが、情報制御フィールドによって示されるレイヤへ何回ループバックすべきかを示すループフィールド回数値をさらに含むことを特徴とする請求項１に記載の方法。

【請求項３】　画像のアニメーション化シーケンスの方法であって、前記画像が単一のマルチレイヤファイル中に含まれており、

ａ）前記画像ファイル中に複数の制御ブロックを提供するステップであって、各制御ブロックは少なくとも１つの前記画像に関連しており、各制御ブロックは、前記画像のいずれの１つが次のシーケンスにあるかを示す情報制御フィールドによって特徴付けられ、少なくとも１つの制御ブロックはシーケンス中の前の画像を示す情報制御フィールドを有し、

ｂ）現在の制御ブロックから現在の情報制御フィールドを読み取るステップと、

ｃ）現在の制御ブロックに関連した画像を表示するステップと、

ｄ）現在の情報制御フィールドが前の画像へループバックすべきことを示す場合には、前の画像の制御ブロックを現在の制御ブロックとして取り入れ、そうでない場合には、次の画像の制御ブロックを現在の制御ブロックとしてシーケンス中に取り入れるステップと、

ｅ）ステップｂ）からステップｅ）を繰り返すステップとを含むことを特徴とする方法。

【請求項４】　複数の画像の１つまたは複数のアニメーションを提供する方法であって、

前記複数の画像を第１の順序で格納するステップと、

前記アニメーションの開始画像を決定するステップと、

開始画像の開始アドレスを決定するステップと、

前記開始画像で開始する、前記１つまたは複数の画像のアニメーション順序を確立するステップと、

前記開始アドレスに向けられる相対アドレス指定を使用して、前記アニメーション順序に依存して１つまたは複数の画像をアニメーション化するステップと、

2

前記少なくとも１つの画像がアニメーション順序で複数回実行される場合には、前記１つまたは複数の画像の少なくとも１つを再使用するステップを含むことを特徴とする方法。

【請求項５】　（ｉ）第１の複数の画像レイヤおよび（ｉｉ）第２の複数の制御ブロックを含んでいるマルチレイヤ画像ファイルを処理する方法であって、前記処理はアニメーションシーケンスを生成し、

対応する制御ブロックに従って画像レイヤを処理し、それによって前記アニメーションシーケンスのための画像を提供するステップと、

画像シーケンス中で画像レイヤが再度使用されるべきである場合には、画像レイヤに再処理のためにタグ付けするステップであって、前記タグ付けは、前記対応する制御ブロックのアドレスに向けられる相対アドレス指定を使用するステップと、

次の制御ブロックに従って前記画像レイヤを再使用し、それによって、前記相対アドレスが前記対応する制御ブロックのアドレスの次のアドレスである場合には、次の画像をアニメーションシーケンスのために提供するステップと、

前記次の制御ブロックに従って次の画像レイヤを使用し、それによって、前記相対アドレスが前記次のアドレスの後のアドレスである場合には、アニメーションシーケンスのために前記次の画像を提供するステップとを含むことを特徴とする方法。

【請求項６】　（ｉ）第１の複数の画像レイヤおよび（ｉｉ）第２の複数の制御ブロックを含んでいるマルチレイヤ画像ファイルを処理する方法であって、前記処理はアニメーションシーケンスを生成し、

対応する制御ブロックに従って画像レイヤを処理し、それによって前記アニメーションシーケンスのための画像を提供するステップと、

画像シーケンス中で画像レイヤが再度使用されるべきである場合には、画像レイヤを再処理のためにタグ付けするステップであって、前記タグ付けは、前記対応する制御ブロックのアドレスに向けられる相対アドレス指定を使用するステップと、

後の制御ブロックに従って前記画像レイヤを再使用し、それによって、前記相対アドレスが前記対応する制御ブロックのアドレスに向けられる後のアドレスである場合には、後の画像をアニメーションシーケンスのために提供するステップと、

前記次の制御ブロックに従って次の画像レイヤを使用し、それによって、前記相対アドレスが前記次のアドレスの後のアドレスである場合には、アニメーションシーケンスのために次の画像を提供するステップとを含むことを特徴とする方法。

【請求項７】　前記処理、タグ付け、再使用、使用ステップが、第１のループ内で第１の回数だけ繰り返し実行

され、前記第１の数は「繰り返し」パラメータに依存していることを特徴とする請求項５に記載の方法。

【請求項８】　前記第１のループが第２のループ内で第２の回数だけ繰り返し実行され、前記第２の数は「ループ」パラメータに依存していることを特徴とする請求項７に記載の方法。

【請求項９】　前記アニメーションシーケンスのための前記画像の提供と前記次の画像の提供との間の時間間隔が、「寿命」パラメータによって実質上決定されていることを特徴とする請求項５に記載の方法。

【請求項１０】　スクリーンにレンダリングされる前記画像のピクセルが、「持続」パラメータに依存してスクリーン上で持続することを特徴とする請求項５に記載の方法。

【請求項１１】　（ｉ）第１の複数の画像レイヤおよび（ｉｉ）第２の複数の制御ブロックを含んでいるマルチレイヤ画像ファイルを処理する装置であって、前記処理はアニメーションシーケンスを生成し、

対応する制御ブロックに従って画像レイヤを処理し、それにより前記アニメーションシーケンスのための画像を提供する処理手段と、

画像レイヤが画像シーケンス中で再度使用されることになっている場合には、再処理のために画像レイヤにタグ付けをするタグ付けし、前記タグ付けは、前記対応する制御ブロックのアドレスに向けられる相対アドレスを使用するタグ付け手段と、

次の制御ブロックに従って前記画像レイヤを再使用し、それにより前記相対アドレスが前記対応する制御ブロックのアドレスの次のアドレスである場合、次の画像をアニメーションシーケンスのために提供する再使用手段とを備えることを特徴とする装置。

【請求項１２】　複数の画像の１つまたは複数の画像をアニメーション化する装置であって、

前記複数の画像を第１の順序で格納するためのファイル構造手段と、

前記１つまたは複数の画像の開始画像の開始アドレスを決定するアンカーアドレス手段と、

前記開始アドレスに向けられる相対アドレス指定を使用して、前記１つまたは複数の画像のためのアニメーション順序を確立するための命令セット手段と、前記１つまたは複数の画像のアニメーションを前記アニメーション順序で提供するアニメーション手段と、

前記少なくとも１つの画像が前記アニメーション順序で複数回実行される場合に、１つまたは複数の画像の少なくとも１つの画像を再使用する画像再使用手段とを備えることを特徴とする装置。

【請求項１３】　アニメーションのために符号化されたマルチレイヤ画像ファイルであって、

第１の複数の画像レイヤと、

第２の複数の制御ブロックとを含んでおり、画像レイヤは対応する制御ブロックに従って処理され、それによって前記アニメーションシーケンスのための画像を提供し、画像レイヤは、画像レイヤが画像シーケンス中で再度使用されるべきものである場合には、再処理のためにタグ付けされ、前記タグ付けは前記対応する制御ブロックのアドレスに向けられる相対アドレス指定を使用することを特徴とするマルチレイヤ画像ファイル。

【請求項１４】　マルチレイヤ画像ファイルであって、

（ａ）第１の順序で格納されている複数の画像と、

（ｂ）実行のための第１のアニメーション命令とを含み、

前記第１のアニメーション命令は、

（ｉ）前記複数のアニメーション化されるべき画像の開始画像の開始アドレスと、

（ｉｉ）前記開始画像の少なくとも１つのアニメーション属性とを含み、

（ｃ）実行されるべき少なくとも１つの次のアニメーション命令であって、前記第１のアニメーション命令および前記少なくとも１つの次のアニメーション命令が連続した順序で実行され、各前記少なくとも１つの次のアニメーション命令は、

（ｉ）アニメーション化されるべき前記複数の画像の次の画像の相対アドレスであって、前記相対アドレスは、前記開始アドレスおよび先行する相対アドレスのいずれかに向けられる相対アドレスと、

（ｉｉ）前記次の画像の少なくとも１つのアニメーション属性とを含むことを特徴とするマルチレイヤ画像ファイル。

【請求項１５】　前記少なくとも１つのアニメーション属性は、

現在の命令の実行の完了と次の命令の実行の完了との間の目標時間間隔を示す「寿命」の値と、

現在の命令の実行の結果として、スクリーンにレンダリングされたピクセルが、表示背景上で持続するように現れるか、実行前の背景にリセットされるかを示す「持続」の値と、

現在の画像を再使用する前に実行する命令の数を示す「次」の値と、

現在の画像を切り取ることなしに表示区域内に配置すべき位置を示す「位置」の値と、

表示区域内に置くために現在の画像に適用されるべき切り取りファクタを示す「サイズ」の値と、

作用を及ぼされるべき画像から切り取る領域を示す「切り取り」の値のいずれかであることを特徴とする請求項１４に記載のマルチレイヤ画像。

【請求項１６】　「次の」値のための零の値が、現在の画像が再使用されないことを示すことを特徴とする請求項１５に記載のマルチレイヤ画像。

【請求項１７】　（ｉ）第１の複数の画像レイヤおよび（ｉｉ）第２の複数の制御ブロックを含むマルチレイヤ

画像ファイルを処理する装置のためのプログラムを格納するためのコンピュータ可読メモリ媒体であって、前記処理はアニメーションシーケンスを生成し、前記プログラムは、
関連する制御ブロックに従って画像レイヤを処理する処理ステップのためのコードであって、それによって前記アニメーションシーケンスのための画像を提供するコードと、
画像シーケンス中で画像レイヤが再度使用されるべきである場合には、再処理のために画像レイヤにタグ付けするタグ付けステップのためのコードであって、前記タグ付けは、前記対応する制御ブロックのアドレスに向けられる相対アドレスを使用するコードとを含むことを特徴とするメモリ媒体。

【請求項１８】　前記プログラムはさらに、
次の制御ブロックに従って前記画像レイヤを再使用する再使用ステップのためのコードであって、それによって、前記相対アドレスが前記対応する制御ブロックのアドレスの次のアドレスである場合には、アニメーションシーケンスのために次の画像を提供するコードと、
前記次の制御ブロックに従って次の画像レイヤを使用する使用ステップのためのコードであって、それによって、前記相対アドレスが前記次のアドレスに続くアドレスである場合には、前記次の画像をアニメーションシーケンスのために提供するコードとを含むことを特徴とする請求項１７に記載のコンピュータ可読メモリ媒体。

【請求項１９】　前記プログラムはさらに、
後の制御ブロックに従って前記画像レイヤを再使用する再使用ステップのためのコードであって、それによって、前記相対アドレスが前記対応する制御ブロックのアドレスに向けられる後のアドレスである場合には、アニメーションシーケンスのために後の画像を提供するコードと、
次の制御ブロックに従って次の画像レイヤを使用する使用ステップのためのコードであって、それによって、前記相対アドレスが次のアドレスに続くアドレスである場合には、アニメーションシーケンスのために次の画像を提供するコードとを含むことを特徴とする請求項１７に記載のコンピュータ可読メモリ媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、マルチレイヤ化画像ファイルフォーマットに関し、特に、アニメーションとして、あるいはマルチレイヤ合成画像のコンテキストで表示されることを意図されているマルチレイヤ画像ファイルに関する。

【０００２】

【従来の技術】マルチレイヤ（またはマルチページ）画像は、典型的には同一のサイズではあるが必ずしもそうではなくともよい１組の画像と考えることができ、出力ディスプレイ装置１１４上に表示されることを目的として、ある理由によって結びつけられているものである。そう考えると、マルチレイヤという表現は、１つのファイルの中の複合画像と言える。ファイル中の各画像はレイヤと称される。現在マルチレイヤ画像が使用される２つの重要なアプリケーション分野があり、それらは、画像編集およびグラフィックデザイン、およびアニメーション、特にインターネット上のウェブページの中のアニメーションを含んでいる。

【０００３】画像編集およびグラフィックデザインの領域においては、マルチレイヤ画像を使用することによって、異なる画像が互いに重ねられて複雑な場面の合成を可能とする。このような場合、通常各レイヤは、それと関連づけられている不透明（またはアルファ）チャネルを有する。様々なレイヤ（画像）を表示装置１１４上に表示するためには、第１のレイヤ（典型的な例としては背景画像）がレンダリングされ、その後にそれに続くレイヤは第１のレイヤ上に合成されるが、たとえばそれは以下の式に従って行われる。

【０００４】

$$Ac = 1\_(1-At)(1\ \ Ab) \quad (1)$$
$$s = At = Ac \quad (2)$$
$$t = (1\_At)\,Ab = Ac \quad (3)$$
$$Rc = sRt + tRb \quad (4)$$
$$Gc = sGt + tGb \quad (5)$$
$$Bc = sBt + tBb \quad (6)$$

上式において、背景画像は、ＲＧＢＡ（赤、緑、青、アルファ）色空間内で、（Ｒｂ、Ｇｂ、Ｂｂ、Ａｂ）として指定され、前景（すなわちトップ）画像は、ＲＧＢＡ色空間内で（Ｒｔ、Ｇｔ、Ｂｔ、Ａｔ）として指定され、出力または合成画像は、ＲＧＢＡ色空間内で（Ｒｃ、Ｇｃ、Ｂｃ、Ａｃ）として指定される。各後続の（または新規な）レイヤは、既存のレイヤと結合（合成）されるまでは、前面画像として扱われ、その後に結合が行われ、（新規の）背景画像となる。このようにして、式（４～６）を、各新規なレイヤに対して、順番に連続的に適用することにより、複数のレイヤを結合して、最終的な合成画像を形成することが可能となる。上述したように式（１）から（６）が最も一般的に使用されているとはいえ、他の結合操作もまた可能である。

【０００５】上述のマルチレイヤ画像の、他の注目すべき応用分野は、アニメーションである。この目的のために、現在で最も広く使用されているファイルフォーマットは、画像交換フォーマット（ＧＩＦ）である。ＧＩＦもまた、連続的な順序に従って合成されているレイヤ（または複合画像）を含む。ＧＩＦファイルの各レイヤは、異なるサイズであってよく、１つのレイヤから次のレイヤへの変化が小面積の領域のみである場合に、記憶容量効率を向上させるために、オフセット座標を使用して配置される。ＧＩＦ標準は、各レイヤが合成される仮

想スクリーンを定義する。それは制御ブロック構造を使用して、ファイル中のレイヤがどのように表示されるべきかを示す。ファイルフォーマットの各レイヤは、制御ブロックの後に続き、以下のものを含んでいる。すなわち、仮想スクリーン中での左上コーナの位置に関する情報、ファイル中の次のレイヤに先立って、そのレイヤがどのくらいの時間表示されるべきかの情報、ファイル中の次のレイヤの表示に先立ってそのレイヤが除去されるべきか否かの情報である。この（制御ブロックに基準をおいた）構造があることによって、デコーダのソフトウェアの実行を取りわけ単純にすることができる。事実、マルチレイヤアニメーションＧＩＦ画像を、正しく表示するＧＩＦデコーダを実行するために必要とされる、付加的な符号化は、非常に小さいものが要求されるにすぎない。

【０００６】ＧＩＦによって使用されるアニメーション機構は、非常に短時間のうちに広く受け入れられるところとなっている。その最も重要な理由は、単純で限定的な設計であるということである。これらの特徴によって、数多くの独立した開発者が、ＧＩＦアニメーションを取り扱うことのできるファイルビューアを、容易に実現できるのである。しかしながら、ＧＩＦの単純であるということが、符号化の効率上の犠牲を強いることになる。たとえば、アニメーションＧＩＦファイル中の各レイヤは、単一の表示フレームに対応しているため、スプライトおよびオーバーレイを使用するアニメーションの符号化が効率的に行えないということが起こる。このことは、各フレームが、分離された画像レイヤとして存在しなくてはならない、ということに由来する。アニメーションの実行中に再使用される画像は、その画像が現れる各フレームに対して、ファイル中に一度だけ現れなければならない。

【０００７】

【発明が解決しようとする課題】より最近になって、未だ開発途上ではあるが、Ｍｕｌｔｉｐｌｅ Ｉｍａｇｅ Ｎｅｔｗｏｒｋ Ｇｒａｐｈｉｃｓ（ＭＮＧ）ファイルフォーマットが、この問題に対処する試みをしている。ＭＮＧは、Ｐｏｒｔａｂｌｅ ＮｅｔｗｏｒｋＧｒａｐｈｉｃｓ（ＰＮＧ）ファイルフォーマットに対する拡張版を基礎としたアニメーションフレームワークを定義している。しかしながら、ＭＮＧではレイヤの再使用が可能とはいえ、ＧＩＦの成功を特徴的に示す単純性の多くは失われている。さらに、アニメーションを記述するためにＭＮＧによる方法は、本質的に実現モデルに結びつくことはない。このことで、ＭＮＧアニメーションのためのビューアの開発が、大幅に実現困難になっている。この問題に対処することの助けとなるために、ＭＮＧの開発者は、ＭＮＧ標準の完全版の複雑性を緩和したサブセット、およびさらに大幅に複雑性を緩和したサブセットを提案した。しかしながら、この問題は、この複雑性緩和サブセットは、ＧＩＦより機能性の向上はほとんどなく、ＧＩＦと同じ符号化効率性の問題を有している。

【０００８】

【課題を解決するための手段】本発明の目的は、従来の工夫の１つまたは複数の欠点を実質的に克服する、または少なくとも改良することである。本発明の第１の態様によれば、（ｉ）第１の複数の画像レイヤおよび（ｉｉ）第２の複数の制御ブロックを含んでいるマルチレイヤ画像ファイルを処理する方法であって、前記処理はアニメーションシーケンスを生成し、前記画像ファイル中に複数の制御ブロックを提供するステップであって、各制御ブロックは少なくとも１つの前記画像レイヤに関連しており、各制御ブロックは、前記制御ブロックおよび関連する画像レイヤのいずれか１つがループバックすべきかを示す情報制御フィールドによって特徴付けられ、各制御ブロックを連続的に実行し、前記情報制御フィールドによって提供される指示に従って、前記実行シーケンス中の前の制御ブロックおよび関連するレイヤへループバックするステップとを含むことを特徴とする方法が提供される。

【０００９】本発明の他の１つの態様によれば、（ｉ）第１の複数の画像レイヤおよび（ｉｉ）第２の複数の制御ブロックを含んでいるマルチレイヤ画像ファイルを処理する装置であって、前記処理はアニメーションシーケンスを生成し、対応する制御ブロックに従って画像レイヤを処理し、それにより前記アニメーションシーケンスのための画像を提供する処理手段と、画像レイヤが画像シーケンス中で再度使用されることになっている場合には、再処理のために画像レイヤにタグ付けをするタグ付けし、前記タグ付けは、前記対応する制御ブロックのアドレスに向けられる相対アドレスを使用するタグ付け手段と、次の制御ブロックに従って前記画像レイヤを再使用し、それにより前記相対アドレスが前記対応する制御ブロックのアドレスの次のアドレスである場合、次の画像をアニメーションシーケンスのために提供する再使用手段とを備えることを特徴とする装置が提供される。

【００１０】本発明の他の１つの態様によれば、アニメーションのために符号化されたマルチレイヤ画像ファイルであって、第１の複数の画像レイヤと、第２の複数の制御ブロックとを含んでおり、画像レイヤは対応する制御ブロックに従って処理され、それによって前記アニメーションシーケンスのための画像を提供し、画像レイヤは、画像レイヤが画像シーケンス中で再度使用されるべきものである場合には、再処理のためにタグ付けされ、前記タグ付けは前記対応する制御ブロックのアドレスに向けられる相対アドレス指定を使用することを特徴とするマルチレイヤ画像ファイルが提供される。

【００１１】本発明の他の１つの態様によれば、（ｉ）第１の複数の画像レイヤおよび（ｉｉ）第２の複数の制御ブロックを含むマルチレイヤ画像ファイルを処理する

装置のためのプログラムを格納するためのコンピュータ可読メモリ媒体であって、前記処理はアニメーションシーケンスを生成し、前記プログラムは、関連する制御ブロックに従って画像レイヤを処理する処理ステップのためのコードであって、それによって前記アニメーションシーケンスのための画像を提供するコードと、画像シーケンス中で画像レイヤが再度使用されるべきである場合には、再処理のために画像レイヤにタグ付けするタグ付けステップのためのコードであって、前記タグ付けは、前記対応する制御ブロックのアドレスに向けられる相対アドレスを使用するコードとを含むことを特徴とするメモリ媒体が提供される。

【００１２】

【発明の実施の形態】付属する図中の１つまたは複数において、同一の数字を有するステップおよび／または特徴に対して参照が行われる場合、それらのステップおよび／または特徴は、この明細書の目的に対して、同一の機能または操作を有するものとする。ただし相反する意図が示された場合はこの限りではない。

【００１３】１つの本発明の実施形態は、たとえば、図１に示されているような汎用のコンピュータを使用して実行できることが好ましく、そのコンピュータにおいては、図２から図８、および図１２から図１９の処理がコンピュータ上でソフトウェアで実行されるように実現されてもよい。特に符号化、復号化法のステップは、コンピュータによって実行されるソフトウェア中の命令によってなされる。また、ディジタル画像を表現するコードストリームの構造の信号を提供するための符号化アルゴリズムも、コンピュータによって実行されるソフトウェア中の命令によって実現されてもよい。ソフトウェアは、例として後述する記憶装置を含む、コンピュータ可読媒体中、に格納されてもよい。ソフトウェアは、コンピュータ可読媒体よりコンピュータにロードされ、その後にコンピュータによって実行される。このようなソフトウェアを有するコンピュータ可読媒体、またはその媒体上に記録されているコンピュータプログラムは、コンピュータプログラム製品である。コンピュータでコンピュータプログラム製品を使用することは、本発明の実施形態に従って、ディジタル画像の符号化、ディジタル画像の構造的符号化表現を復号または信号化するために、好ましく有利な装置をもたらす。

【００１４】コンピュータシステム１００は、コンピュータ１０１、ビデオディスプレイ１１４、入力装置１０２、１０３からなる。さらに、コンピュータシステム１００は、複数のラインプリンタ、レーザプリンタ、プロッタ、コンピュータ１０１に接続されているその他の再生装置を含む他の出力装置１１５のうちの、いずれをも有することができる。コンピュータシステム１００は、モデム１１６、コンピュータネットワーク１２０またはそれに類するものを介して、適当な通信チャネルを使用して、１つまたは複数の他のコンピュータへ接続されることができる。コンピュータネットワークは、ローカル区域ネットワーク（ＬＡＮ）、ワイド区域ネットワーク（ＷＡＮ）、イントラネットおよび／またはインターネットを含んでいてもよい。

【００１５】コンピュータ１０１そのものは、図１中の中央演算装置（１つまたはそれ以上）（以降単純にプロセッサと称する）１０５、ランダムアクセスメモリ（ＲＡＭ）およびリードオンリーメモリ（ＲＯＭ）を含むメモリ１０６、入出力（ＩＯ）インターフェース１０８、ビデオインターフェース１０７、一般的に図１でブロック１０９で表される１つまたは複数の記憶装置からなっている。記憶装置１０９は、１つまたは複数の、フロッピー（登録商標）ディスク１１１、ハードディスクドライブ１１０、光磁気ディスクドライブ、ＣＤ－ＲＯＭ、磁気テープまたは、当技術分野に長じた技術者にとってよく知られている複数の不揮発記憶装置からなる。コンポーネント１０５から１１３までの各々は、一般的には、そのものはデータ、アドレス、制御バスからなるバス１０４を介して、１つまたは複数の他の装置に接続されている。

【００１６】ビデオインターフェース１０７はビデオディスプレイ１１４に接続されており、コンピュータ１０１から、ビデオディスプレイ１１４上の表示のために、ビデオ信号を提供する。コンピュータ１０１を操作するためのユーザ入力は、１つまたは複数の入力装置によって供給される。たとえば、オペレータは、キーボード１０２および／またはマウス１０３のようなポインティングデバイスを使用して、コンピュータ１０１に対して入力供給することができる。

【００１７】システム１００は、例証の目的で提供されているのみであり、本発明の領域および趣旨から逸脱することなく他の機器構成を採用することが可能である。実施形態が実行できるコンピュータの例としては、ＩＢＭ－ＰＣ／ＡＴまたはその互換機および、マッキントッシュ（ＴＭ）ファミリーのＰＣ、サンスパークステーション（ＴＭ）または類似したもののうちの１つを含む。前述のコンピュータは、単に本発明の実施形態が実行可能なコンピュータのタイプの例示にすぎない。典型的な例としては、以下に記述する実施形態のプロセスは、コンピュータ可読媒体である（図１において一般的にブロック１１０として表現されている）ハードディスクドライブ上に記録されているソフトウェアまたはプログラムとして存在し、プロセッサ１０５を使用して読み取りがなされ、制御される。プログラム、ピクセルデータおよび、ネットワークより取り込んだデータはどのようなものでも、それらの中間的な格納は、可能であればハードディスクドライブ１１０と呼応して動作する半導体メモリ１０６を使用して行われることができる。

【００１８】ある例においては、プログラムはユーザに

対して、ＣＤ－ＲＯＭ、フロッピーディスク（ともにブロック１０９として一般的に表現されている）において符号化されて供給されるか、またはその代わりに、たとえばコンピュータに接続されているモデム装置を介してネットワークからユーザが読み取ることができる。さらにまたソフトウェアは、磁気テープ、ＲＯＭすなわち集積回路、光磁気ディスク、無線またはコンピュータと他の装置間の赤外線通信チャネル、コンピュータが読み取ることができるＰＣＭＣＩＡカードのようなカード、電子メール伝送およびウェブサイトおよびそれに類するものに記録されている情報を含むインターネットとイントラネット、を含む他のコンピュータ可読媒体から、コンピュータシステム１００に、ロードされることも可能である。前述の例は、関連したコンピュータの読み取ることができる媒体の単なる例示にすぎない。他のコンピュータの読み取ることができる媒体は、本発明の領域および趣旨から逸脱することなく使用することができる。

【００１９】あるいは符号化方法の実施形態は、符号化、復号、信号化プロセスの機能またはサブ機能を実行する１つまたは複数の集積回路のような専用のハードウェア内で実行されることができる。この専用ハードウェアは、ＡＳＩＣおよび連結オンチップメモリを含む。

【００２０】［第１の実施形態］２つのマルチレイヤ画像ファイルフォーマットについて図２および図３を参照しながら説明する。

【００２１】図２および図３に例示された各々には、デ＊コーダに対して、レイヤが表示される出力表示装置１１４（図１参照）の区域のサイズを示す、ヘッダ情報２１０の集合を含むファイルの開始部分が示されている。ファイル中の各レイヤに対して、関連づけされた制御ブロック２２０がある。図２および図３に示された「ＣＢ」は数字Ｎだけ続く。数字Ｎは、Ｎ番目の制御ブロックおよびＮ番目のレイヤ間の対応を示す。たとえば、「レイヤ２」と関連づけられている制御ブロックは「ＣＢ２」と表現される。制御ブロックは図２に示すようにファイルのヘッダ領域に集められており、そのような場合には、その順序は、ファイル中に含まれる画像レイヤのシーケンス順序に対応している。一方図３に示すように、制御ブロックは、ファイル中の各画像レイヤに先行して配置されている。画像レイヤは、標準的なまたは専用の符号化体系を使用して符号化されるか、または未処理のピクセル値の連続として単純に表現されていてもよい。たとえば、ファイルフォーマット中の各レイヤは、ウエーブレット準拠符号化体系を使用して独立して符号化されるか、またはＪＰＥＧ（Ｊｏｉｎｔ　Ｐｉｃｔｕｒｅ　Ｅｘｐｅｒｔ　Ｇｒｏｕｐ）圧縮法を各レイヤに対して使用することもできる。

【００２２】表１は、複数の情報フィールド、およびそれらの関連する説明であり、本発明の実施形態で使用されるものである。

【００２３】

【表１】

【表１】

フィールド	説明
x	出力装置の表示エリアの左端から、対応するレイヤの左端までのピクセルずれ
y	出力装置の表示エリアの上端から、対応するレイヤの上端までのずれ
t	次のレイヤを表示するに先立ち、現在のレイヤのレンダリング後に、その表示を保持する時間長
r	次のレイヤを表示するに先立ち、現在のレイヤを除去する（そして表示を以前の状態に復帰させる）かどうかの指示
b	ｎの条件に従ってレンダリングする次のレイヤを示す
n	ファイル中のレイヤを表示するに先立ち、ｂによって定義されるブランチの後に続く回数を示し、ファイル中の次のレイヤを連続的に次にレンダリングすべきものとして扱う

【００２４】各制御ブロックは複数の情報フィールドを含み、各フィールドは制御ブロックに対応するレイヤに関する情報を提供する。表１においては、本発明の実施形態に従う複数の情報フィールド、および示されている各フィールドのための対応する説明の概略が示されている。各フィールドは、最上位バイトが最初に出現するように格納されている、３２ビット符号なし整数を使用したファイルの形式で表現されている。本発明の趣旨を逸脱することなしに、可変長表現法を含む他の表現法を、必要に応じて使用することもできる。

【００２５】表１の最初の２つのフィールド、ｘおよびｙについて、図４を参照しながらさらに説明する。ｘおよびｙは、対応する画像レイヤ４１０の左上コーナの画像を表示するために使用されるスクリーン区域４２０の左上コーナからのずれを示す。各軸の表現形式は図４に示すとおりである。

【００２６】第３のフィールドの、ｔは次のレイヤを表示するに先立ち、現在のレイヤのレンダリング後にその表示を保持すべき時間を示す。このフィールドの単位はたとえば、１００分の１秒である。したがって、ａ＋３

の値は、１００分の３秒を表す。一方、他のメディアとの同期が要求される場合には、その値はタイマの刻み（ｔｉｃｋ）の１単位となる。タイマの１刻みとは、任意の（システムによって定義される）単位時間である。刻みはしばしば、分離したオーディオまたはビデオデータファイルの１部を形成する同期「トラック」から導かれる。したがって、この代替表現での「ｔ」値の３は、同期クロックの３刻みを表現する。

【００２７】第４のフィールドは、次のレイヤの表示に先立ち、対応するレイヤを除去する（そして表示を以前の状態に復帰させる）かどうかを示す。このフィールドは、論理フィールドであって、出力表示装置１１４が、それぞれの以前の状態に復帰させるかどうかに依存して、「真」または「偽」値が割り当てられている。現在のレイヤは除去すべき場合、ｒ＝真であれば、デコーダは現在のレイヤを表示するに先立って、出力表示装置１１４の状態を格納しなくてはならない。逆に、レイヤが除去されるべきでない場合、ｒ＝偽であれば、後続のレイヤは、（これまでに表示されたすべての）レイヤの上に合成される。

【００２８】表１の第５および第６のフィールド（それぞれｂおよびｎ）はループ処理機構を実現する。第５（ｂ）のフィールドは、オプションで枝分かれする行き先レイヤを示す。これは、零より大きいか等しい数字で表現され、次のレイヤに関連しているファイル中で戻るべきレイヤの数として解釈される。この定義は、ファイル中の次のレイヤ、すなわち最も「自然な」進行を示すｂ＝０の値となる。ｂ＝１の値は現在のレイヤ、すなわち現在ｔフィールドを使用して一般的に特定されることのできるものを越えている、対応するレイヤの表示時間を延長するために使用される特徴を示す。このようにして、ｂ＝３の場合は、表示されるべきレイヤの順序は、現在のレイヤの前に２レイヤ分だけループバックし、それらのレイヤを再表示しなければならないということを示す。このことは、現在のレイヤを含んで最後の３つのレイヤを繰り返すということと同じである。ｂが２３２であるか、または現在表示されている（ファイル中の）レイヤの連続番号＋１よりも大きい値のどれかである場合、枝分かれしてゆくべきレイヤは、ファイル中の第１のレイヤとして定義される。

【００２９】表１の第６のフィールドのｎは、第５のフィールドで示される枝分かれの回数を示すものであるが、ファイル順序中の次のレイヤの表示に先立って行われなければならない。ｎの値ｎ＝２３２は、枝分かれの実行は無限であること（ユーザまたは他のより高度のレベルの制御によって意図的に停止されるまでは）を示すために使用される。プログラミング技術に通じた者であれば、２３２は３２ビット整数としての表現に従う場合に、ｎが取ることができる最大の値であることを理解するであろう。他の特定の値は、本発明の趣旨から逸脱することなく使用することができる。

【００３０】図５は、本発明の実施形態に従うマルチレイヤー画像を表示するプロセスステップの流れ図を示す。プロセスはブロック５００から開始される。ファイルの画像ファイルヘッダは、他の情報の中でレイヤの合計数についての情報を含むブロック５１０で読みこまれる。これによって、ステップ５１５において、画像データが表示される表示装置１１４のために、表示区域が決定されて初期化されるようになる。メイン表示ループはブロック５２０において、現在のレイヤが、ファイル中の第１のレイヤを示す値である１に初期化されることを表示する変数の値で初期化される。表示ループはブロック５３０において、現在のレイヤのための関連する制御ブロックを読み込むこと、および現在のレイヤのための画像データを読み込むことで開始される。ブロック５４０において、現在のレイヤのための画像データが、その関連する制御ブロックからの情報によって指定されるように表示される。その後に現在のレイヤとして記録される値は、ブロック５５０で更新される。このステップの目的は、表示ループの次の実行において表示されるべきレイヤのインデックスを確定することである。これに先立ち、現在のレイヤの新たな値として決定されるべき値は、それがファイル中に存在するレイヤの総数よりも大きいかどうかを決定するために、５６０において判断が行われ、その後にブロック５９９において実行が終了する。すなわち制御ブロック５６０が「真」と判断すれば、表示ループはループを抜けてブロック５９９へ行く。そうでない場合は、制御ブロック５６０が「偽」と判断し、制御は、現在のレイヤのために新たに確定された値を使用してブロック５３０へ帰る。

【００３１】図６は、ブロック５３０で実行される処理ステップ「現在のレイヤのために制御ブロックを読む」をより詳細に説明するものである。処理はブロック８００から開始される。判定ブロック８１０において、これがこのレイヤの制御ブロックが読まれた１回目（したがってこのレイヤが表示された１回目）であるかどうかを決定するための判断が行われる。制御ブロック８１０が「偽」と判断すれば、それは今回がこのブロックが読まれた１回目であること、およびそのレイヤがこれまで表示されていなかったことを示し、その制御ブロックのための実際の制御パラメータが８３０において読まれ、処理がブロック８９９へ抜ける前に、変数ｃがそのレイヤに対して裏付けされて、ブロック８２０で零にセットされるこの変数ｃは、図８に示されている流れ図に関連して後述される後のループ計算に使用される。一方判定ブロック８１０が「真」と判断すれば、それはそのブロックが過去に読まれたことがあり、表示されたことがあるということを示し、過去に読まれたその制御ブロックのためのパラメータが８４０で読み込まれて、処理はブロック８９９へと抜ける。ブロック８１０が真の時は、変

数ｃの値は、前のループから開始ブロック８００へ入った時のものと同じ値に保持される。

【００３２】図７については、図５の表示レイヤブロック５４０のプロセスステップのより詳しい内容を示している。レイヤ表示プロセスステップはステップ６００から開始される。判定ブロック６１０において、現在のレイヤのための制御ブロックからのパラメータｒ（表１の情報フィールド参照）が、値が「真」かどうかの判定が行われる。判定ブロック６１０が「真」と判断すれば、現在のレイヤが表示されるときオーバーライトされる表示区域がブロック６２０で表示外格納区域に格納される。レイヤはその後にブロック６３０で、表示区域上で（そのレイヤの制御ブロックからの、および図４で表されているように）パラメータｘおよびｙによって特定される点に合成される。その後、ブロック６４０において、表示されている画像の現在の状態が、（そのレイヤの制御ブロックからの）パラメータｔで特定される時間長の間保持される。格納されたスクリーン区域はその後、ブロック６９９で抜ける前に、ブロック６５０で、表示外格納区域から復帰される。一方、判定ブロック６１０が「偽」と判断すれば、格納、復帰操作は要求されない。レイヤはブロック６３０で、表示区域上で（そのレイヤの制御ブロックからの、および図４で表されているように）パラメータｘおよびｙによって特定される点に合成される。その後、ブロック６４０において、表示されている画像の現在の状態が、ブロック６９９で抜ける前に（そのレイヤの制御ブロックからの）パラメータｔで特定される時間長の間保持される。処理はその後、図５の次の実行ブロック５５０「現在のレイヤの値の更新」で続行する。

【００３３】新たな現在のレイヤ変数（図５のブロック５５０）の計算において行われる処理は、図８を参照して説明する。プロセスはブロック７００で開始される。判定ブロック７１０において、パラメータｂ（現在のレイヤの制御ブロック）が、値零と比較される。値零は表示されるべき次のレイヤは、ファイル中の次のレイヤであることを示す。判定ブロック７１０が「真」と判断すれば、現在のレイヤの値はブロック７８０で１だけ増分されて、処理はブロック７９９で抜ける。そうではなく、判定ブロック７１０が「偽」と判断すれば、判定ブロック７２０で、現在のレイヤの値である変数ｃが、「１」と比較される。制御ブロック７２０が真と判断すれば、現在のレイヤの変数の値は、ブロック７５０で零にセットされ、現在のレイヤの値は、ブロック７８０で増分され、ブロック７９９へ処理は抜ける。そうではなく、制御ブロック７２０が「偽」と判断すれば、レイヤのパラメータｃの値は、制御ブロック７３０で、値２３２と比較される。制御ブロック７３０が「真」と判断すれば、現在のレイヤは、ブロック７９０で現在のレイヤ＋１－ｂの値にセットされ、処理はブロック７９９へ抜ける。そうではなく、制御ブロック７３０が「偽」と判断すれば、レイヤのパラメータｃの値は、制御ブロック７４０で、値零と比較される。制御ブロック７４０が「真」と判断すれば、ｃの値は、ブロック７６０で（レイヤの制御ブロックからの）現在のレイヤのパラメータｎの値に等しくセットされる。その後現在のレイヤは、ブロック７９０で、現在のレイヤ＋１－ｂの値にセットされ、処理はブロック７９９へ抜ける。そうではなく、制御ブロック７４０が「偽」と判断すれば、現在のレイヤのパラメータｃの値はブロック７７０で減分される。その後に、現在のレイヤは、ブロック７９０で、現在のレイヤ＋１－ｂの値にセットされ、処理はブロック７９９へ抜ける。

【００３４】［第２の実施形態］図９は本発明の実施形態の画像ファイル構造を示す。ファイル１０００は、バイナリファイルの中に、連続的にパックされた１００２から１００８までの多くのエレメントを含んでいる。ファイルの始めの部分のエレメントは、ヘッダ情報１００２を含み、それはファイルタイプを識別する情報、ならびにファイル１０００に含まれる画像データのパラメータを記述する情報をも含む。エレメントはまた、ファイルリーダのすべてが理解する必要はない基本的なファイル構文の拡張をも記述してもよい。

【００３５】本実施形態においては、各命令は同一のパラメータ化法を有しており、結果として、固定長を採用する。この事実は、ファイルのリーダが使用して、命令の境界、および命令セットの長さが既知の場合には、命令の数を決定する。アニメーション制御ブロック１００４は、ブロックが埋め込まれているファイル１０００の構文を使用する。通常このことで、ファイルリーダが、制御ブロック１００４の開始ポイントおよび全体の長さを決定することができる機構が提供される。各命令セット、たとえば１０２０（先導の繰り返しパラメータ１０２８を含む）は、ファイルリーダが、開始ポイントおよび各セット１０２０の長さを決定できるような方法で、単純明快に区切られる。本実施形態においては、各命令セットは、（ｉ）命令セットの長さを示している符号なし３２ビットの整数、（ｉｉ）後続のデータがアニメーション命令のセットであることを示す４バイトのタグに対して付加される。この構造化機構は、例証であって、各命令セットの開始部のオフセットを記載している表のような異なる構造も同じく使用可能である。

【００３６】ファイル１０００は、画像データ１００６または画像データへの参照を含む、１つまたは複数のエレメントを含む。ファイル中に含まれ参照される、複数の別個の静止画像１００６～１００８があってもよく、それらはレイヤと称される。これらのレイヤの内には、それらはオーバーレイされるか表示の目的でファイル中で他の画像レイヤと合成されることを意図されているため、それが別個に見た場合には不完全であってもよい。

しかしながら、各々は、完全なコードストリームまたはコードストリームの組み合わせであり、別個に復号されることができ、本明細書の領域内で明確なものである。アニメーションは、画像レイヤ１００６～１００８の１つまたは複数を、単一もしくは組み合わせで使用して実行できる。

【００３７】各画像レイヤたとえば１００６は、１つまたは複数のチャネルを備えており、それはファイル１０００中に１つまたは複数のコードストリームとして存在しているか、またはルックアップテーブルを介してファイルによって参照されるかマッピング画像エレメントとして導かれる。ファイル１０００中の各コードストリームまたは参照情報は、１つまたは複数のファイルエレメント中に存在する。ヘッダエレメント中の情報はファイルリーダが使用して、完全なコードストリームを取り出し、それらを復号して画像レイヤとする。

【００３８】各レイヤのチャネルは、ピクセル値の配列を備えている。これらは、カラー空間に特有のカラー情報のサンプルに対応しており、ファイルのヘッダエレメント１００２内で定義されている。１つのチャネルはまた、グレースケール画像での強度に対応していてもよい。１つまたは複数のチャネルはまた、レイヤ中の他のチャネルをレンダリングする際に使用するための、不透明性の情報のサンプルを含んでいてもよい。このチャネルは一般に、アルファチャネルと称される。アルファチャネルデータは、バイナリ（またはｂｉ－レベル）であって、完全な透明および完全な不透明に対応する１つまたは２つの可能な値のみを取る各サンプルを有する。バイナリのアルファデータは、完全に透明なピクセルすべてに対して一意のカラーを割り当てることによって、カラーチャネルとともに符号化される。

【００３９】この仕様は、アニメーションを記述する方法を開示しており、それは、（ｉ）ファイル中に含まれるアニメーションを表示するために必要とされるスクリーン区域（たとえば図１０の１５３２）、（ｉｉ）アニメーション制御情報１００４のブロック、（ｉｉｉ）どのようなものでも適当な方法を使用して符号化された画像レイヤ１００６～１００８のシーケンス、を含むが、それに限定されないグローバルパラメータを有するヘッダ１００２を含むファイルまたはコードストリーム１０００を備えている。

【００４０】アニメーション制御情報１００４（アニメーション制御ブロックとも称する）は、拡大した表示１０１６に示すように、タイマの刻みの継続時間を定義する「刻み」と表示されている整数１０１４を含んでいる。アニメーション制御情報１００４はまた、アニメーションが全体として表示されるべき回数を定義する「ループ」と表現されている整数１０１８を含んでいる。アニメーション制御情報１００４はさらに、１つまたは複数のフレーム制御命令の組１０２０～１０２２を含む。アニメーション制御ブロック１００４の構造は、表２を参照して説明する。

【００４１】

【表２】

【表２】

フィールドタグ	符号化	説明
刻み（tick）	16 ビット符号なし整数	タイミング命令を解読するために使われる、デフォルトのタイマ刻み（tick）のミリ秒での持続時間。
ループ	16 ビット符号なし整数	このアニメーションの完全な表示を繰り返す回数。値 2^{16} は、デコーダはアニメーションを無限に、または外部信号によって停止されるまで、繰り返すことを示す。
命令セット	テーブル２参照	アニメーション命令の集まり

表２．アニメーション制御ブロック中のフィールドの説明

【００４２】「ループ」１０１８の既定値は、アニメーションが無限回数繰り返されることを保証するために、使用することができる。

【００４３】フレーム制御命令の各組１０２０～１０２２は、拡大表示１０３２に示すように、関連した命令セット１０３０～１０３６が実行されるべき回数、およびリーダによって連続的な順序で実行されるべき命令セットを示す、「繰り返し」と表示されている先頭整数１０２８を含む。「繰り返し」の既定値は、アニメーション命令シーケンスが、無限回実行されることを保証するために使用される。表３には、１０２０～１０２２の命令セットの形式を要約している。

【００４４】

【表３】

【表３】

フィールドタグ	符号化	説明
繰り返し	16 ビット符号なし整数	アニメーション命令保証の実行を繰り返す回数
命令 m	表３参照	アニメーション命令

表３．各アニメーション制御ブロックの「命令セット」に含まれるフィールドの説明

【００４５】各命令、たとえば１０３４は、１０４２の拡大した表示（矢印１０６０および１０６２で示したように、これはセクション１０５８および１０６４が直列に並んでいる）に示すように、現在の命令の実行完了および次の命令の実行完了の間に（理想的に）起こるべき、タイマの刻みの数を定義している「寿命」と表示されている整数１０４４を含んでいる。この命令はさらに、現在の命令を実行した結果としてスクリーンにレンダリングされたピクセルが、表示の背景に持続的に現れるべきか、または実行前の背景にリセットされるために現れるべきかを定義する、「持続」と表現されているバイナリ形式のフラグ１０４６を含んでいる。さらに、「次」と表現されている整数１０４８は、値零が、そのレイヤが、零以外の「ループ」制御の結果としてグローバルループが実行されたにもかかわらず、後続の命令のどれによっても再使用されてはならないということを意味する現在のレイヤを再使用する前に実行される命令の数、を定義する。

【００４６】第１の命令１０３０は、ファイル１０００中の第１のレイヤ１００６に作用し、各後続の命令は、前の命令の「次」フィールド中の命令に特定されたレイヤに作用し、またはそのような特定がなされていない場合は、ファイル中の連続した次のレイヤに作用する。

【００４７】「寿命」（すなわち１０４４）に対して零の値、および「持続」（すなわち１０４６）に対しての偽の値は、その命令によって作用を及ぼされるレイヤは、その命令によってどのような方法でもレンダリングされないことを示す。

【００４８】「寿命」（すなわち１０４４）に対して零の値、および「持続」（すなわち１０４６）に対して真の値が得られれば、現在の命令によって作用を及ぼされるレイヤはフレーム定義シーケンスの部分であると見なされる。このようなシーケンスは、「寿命」に対して、零でない値を有する次の命令の実行によって停止される。フレーム定義シーケンスの停止で、結果として、そのフレーム定義シーケンスによって作用を及ぼされるすべてのレイヤの合成および表示が、命令の停止のための「持続」および「寿命」値が一括して適用されるような方法で行われる。表示見解からみれば、フレーム定義シーケンス中のすべての命令は、単一の命令として実行されているように見えるべきである。

【００４９】「寿命」（すなわち１０４４）の値を予め最大に設定することは、この命令の実行の後、アニメーションは無限に停止されることを示す。このような場合には、何らかの、より高度な相互作用があれば続行されることができる。各命令（１０３０）は、この命令によって作用を及ぼされるすべてのレイヤの画像のための表示区域内で、左上コーナを置くための位置を定義する「（ｘ，ｙ）」で表されるペアの整数１０５０、１０５２を付加的に含むことができる。命令１０３０はまた、この命令によって作用を及ぼされるレイヤから切り取る領域の左上部、幅、高さを定義する「（Ｃｘ，Ｃｙ，Ｃｗ，Ｃｈ）」と表される整数のセット１０６６～１０７２をも含むことができる。切り取られる領域は、この命令の範囲内のみで作用を及ぼされるレイヤを置き換えるもの、と見なされる。

【００５０】各命令は、この命令によって作用を及ぼされるレイヤがレンダリングされるべき表示区域内での、領域の幅、高さを定義する「（ｗ，ｈ）」で表される、ペアの整数１０５４、１０５６を付加的に含むことができる。このステップは、レイヤの幅および高さが、命令の中で特定される値と異なる場合に、レイヤを再サンプリングするステップを含む。命令１０３４～１０３６の形式を表４に示す。

【００５１】

【表４】

【表４】

フィールドタグ	好ましい符号化法	説明
持続	1 ビットのフラグ	現在の命令の実行の結果、スクリーンにレンダリングされたピクセルが、命令の寿命が切れた後に、実行前の背景に対して持続するように見えるか、リセットされるように見えるか、を示す論理フラグ
寿命	15 ビットの符号なし整数	この命令の完了および次の命令の完了の間に置くべきタイマの刻み数
次	32 ビット符号なし整数	現在の画像レイヤを再使用する前に、（現在の命令を含む）実行する命令の数。値零は、零でない値の「ループ」の結果としてグローバルループが実行されたにもかかわらず、どのような後続の命令のためにでも、レイヤが再使用されてはならないことを示す。
x_スクリーン	32 ビット符号なし整数	この命令によって作用を及ぼされるレイヤの左端を置くべき、表示エリアの左端からの距離で、スクリーンピクセルで表現される。
y_スクリーン	32 ビット符号なし整数	この命令によって作用を及ぼされるレイヤの上端を置くべき、表示エリアの上端からの距離で、スクリーンピクセルで表現される。
w_スクリーン	32 ビット符号なし整数	この命令によって作用を及ぼされるレイヤを切り取りおよびレンダリングすべき表示エリアの幅で、スクリーンピクセルで表現される。
h_スクリーン	32 ビット符号なし整数	この命令によって作用を及ぼされるレイヤを切り取り、およびレンダリングすべき表示エリアの高さで、スクリーンピクセルで表現される。
x_切り取り	32 ビット符号なし整数	この命令によって作用を及ぼされるレイヤ内で切り取る領域の左端への距離で、画像ピクセルで表現される。
y_切り取り	32 ビット符号なし整数	この命令によって作用を及ぼされるレイヤ内で切り取る領域の上端への距離で、画像ピクセルで表現される。
w_切り取り	32 ビット符号なし整数	この命令によって作用を及ぼされるレイヤ内での切り取り領域の幅で、画像ピクセルで表現される。
h_切り取り	32 ビット符号なし整数	この命令によって作用を及ぼされるレイヤ内での切り取り領域の高さで、画像ピクセルで表現される。

表４．アニメーション制御ブロックの命令フィールドに含まれるフィールドの説明。

【００５２】命令パラメータの解釈を、図１０を参照して更に行う。

【００５３】図１０は、仮想スクリーン１５２６を示し、そこにおいてレイヤ１５０４の切り取りセグメント１５１０がレンダリングされ、レンダリングの実行は矢印１５２０で表されている。仮想スクリーン１５２６は、幅１５２４、高さ１５４０で、これらの寸法は、黒点で表されている（x，y）原点１５２２に基づいている。レイヤ１５０４のセグメント１５１０がレンダリングされる、仮想スクリーン１５２６のセグメント１５３２は、幅１５３０および高さ１５３４を有し、図９でそれぞれ１０５４および１０５６と表されており、これらの寸法は、図９の１０５０、１０５２で表されている（x，y）原点１５２８に基づいている。仮想スクリーン１５２６上へレンダリングされるレイヤ１５０４のセグメント１５１０は、それぞれ図９の１０７０および１０７２で表されている、幅１５１２、および高さ１５１４を有し、これらの寸法は、図９の１０６６、１０６８で表されている（x，y）原点１５０８に基づいている。レイヤ自身１５０４は、幅１５０６および高さ１５１８を有し、これらの寸法は（x，y）原点１５０２に基づいている。

【００５４】図１１は、アニメーション化された画像シーケンスを表示するためのメモリ配置１１００を示す。メモリは、ファイルヘッダ１００２で定義されるスクリーン区域のサイズと等しい容量を有する、可視メモリ領域１１０２、少なくとも１度に（単体命令の、またはフレームの定義シーケンスの最終命令の実行の結果から来る）スクリーンにレンダリングされる、最大区域のサイズに等しい画面外作業領域１１０４を含む。メモリは、さらに画面外補助記憶装置１１０６を含んでおり、これは、少なくとも、１度に（単体命令の、またはフレームの定義シーケンスの最終命令の実行の結果から来る）スクリーンにレンダリングされる、最大区域のサイズに等しい。さらにメモリは、「レイヤメモリ」と称されるレイヤのリストのための記憶装置１１０８を含み、これは

そのリスト内に明示的に記憶されている、どのレイヤの復号された版をも再生させるために使用できる。この装置はさらに、レイヤメモリ内のエントリに関連しており、および（レイヤメモリ中の）関連するレイヤが作用を及ぼされる前に、さらに実行されるべき命令の数、を含んでいる整数のリストのための記憶装置１１１０を備えている。

【００５５】図１２は、アニメーションプロセスの実施形態３７２の、実行のトップレベルを示す。実行はステップ３００より開始される。マルチレイヤファイル（図９の１０００）がステップ３０５でオープンされ、ステップ３１０でヘッダおよびアニメーション制御情報が読み込まれる。これは図１３で詳説される。ヘッダ情報は、表示リソースおよびサポート構造をステップ３１５で割り当てるために使用される。これは図１４で詳説される。メモリおよびサポート構造は、ステップ３２０で初期化される。これは図１５で詳説される。要求された命令の決定は、ステップ３２５で実行される。これは図１６で詳説される。要求された画像レイヤの決定はステップ３３５で実行される。これは図１７で詳説される。命令に従って行われるレイヤのレンダリングは、ステップ３４５で実行される。これは図１８（ａ）および図１８（ｂ）で詳説される。スクリーンへのレンダリングされた長方形の流し込みはステップ３６７で実行される。これは図１９で詳説される。

【００５６】プロセス３７２のメインアニメーションループはステップ３２５で開始される。このステップでは、プレーヤは、アニメーション制御ブロック内で提供されている命令のセットから、使用する表示命令を決定する。決定された命令はステップ３３０で判定され、命令が、それ以上の命令が使用可能でないことを示す特定の値を有するかどうかを決定する。この特定の値は「停止」を表す。「停止」以外の命令が見つけられた場合は、プロセス３７２の実行はステップ３３５へ移動し、プレーヤは命令によって作用を及ぼされるべき画像レイヤを決定する。決定されたレイヤはステップ３４０で判定され、決定されたレイヤが、もはやどのようなレイヤも発見できないことを示す「空」と表される特定の値を有するかどうかを決定する。

【００５７】使用するレイヤがある場合は、プロセス３７２の実行はステップ３４５へ移動し、そこでレイヤは命令に従ってレンダリングされる。後続のステップ３５０では、命令の「次」フィールドが、そのレイヤは再度使用されることを示す、零以外の値と比較される。零以外の値が発見されると、レイヤおよび「次」フィールドの値がステップ３５５でレイヤメモリへ引き渡される。レイヤメモリは、実行目的のために、複数の異なるが、しかし機能的には等価の形式を取ることができる。１つの実施形態では、レイヤメモリの各エントリは、復号された画像サンプルを格納する。他の実施形態では、レイヤメモリ内に、圧縮されたコードストリームが保持される。さらに別の実施形態では、ファイル中のレイヤの最初のバイトへのポインタが、そのデータを読み取り復号するために要求される、どのような補助データとも一緒に格納される。すべてのケースにおいて、レイヤメモリは、将来のために格納されているレイヤのピクセルを、リーダが再生できるように、十分な情報を提供する。

【００５８】現在の命令の「次」フィールド（すなわち１０４８）が零である場合には、このことはこの命令の実行の後には必要とされないことを示し、そのレイヤピクセルデータ、またはデコーダの状態を維持するように適合されたどんなメモリも、すべて解放される。どちらのケースにおいても、プロセス３７２の実行は、次に、後続の命令が決定されるステップ３２５へ帰り、次のレイヤが決定されレンダリングされ、以降同様に続く。

【００５９】もしどのような命令も発見されない場合（すなわち、ステップ３３０が、「命令は「停止」か」という判定で「ｙｅｓ」の値を判断する場合）、または、どのようなレイヤも発見されない場合（すなわち、ステップ３４０が、「レイヤは「空」か」という判定で「ｙｅｓ」の値を判断する場合）には、アニメーションシーケンス３７２は終了しなければならないとされ、実行はステップ３６０へ移行する。

【００６０】ループフィールドの値がステップ３６０で判定されて、零であれば、アニメーションプロセス３７２の実行は停止する。しかしながら、実行された最後の命令が寿命フィールドで零を有していれば、レンダリング画像中にレンダリングされるのを待っている未表示の画像データがある可能性がある。論理的にアニメーション（または代わりに静止合成）を完了するために、レンダリング画像のレンダリング長方形は、ステップ３７０へ抜けるに先立ち、ステップ３６７でスクリーンに流し込まれる。代わりの実施形態において３６７の流し込みステップは、判定ステップ３６０に先立って実行されることができる。ステップ３６０でループフィールドが零でない場合、ループフィールドの値はステップ３６５で減分され、その後に（ステップ３２０で）メモリ、サポート構造の再初期化が行われ、アニメーションループが再開される。

【００６１】図１２のステップ３１０は図１３を参照して詳細に説明する。実行はステップ１６１０より開始される。後続のステップ１６２０において、アニメーションのレンダリングのために使用されるスクリーン区域の幅および高さは、ファイルから画像データのリカバリに重要な他のヘッダ情報と共に読み込まれる。しかしアニメーションプロセスにおいて、幅および高さのパラメータのみは重要な役割を果たす。後続のステップ１６３０において、トップレベルアニメーション制御が読み込まれ、これらは、表１に定義した「刻み」および「ループ」パラメータを含んでいる。次いでステップ１６４０

において、アニメーション命令セットが読み込まれる。実際において、命令セットは完全な形で読み込まれるか、またはメインアニメーションループの実行中に読み込みのために維持されるファイルへのポインタが読み込まれる。図９に関連して説明したヘッダ情報１００２は、本実施形態を実現するために要求されるヘッダ情報の、要求される最小のセットのみを表す。他の実施形態では、任意の付加的なヘッダ情報を組み込んでもよい。最終的にステップ１６５０で実行を抜け出る。

【００６２】図１２のステップ３１５は図１４を参照して詳説される。実行は、ステップ１７１０で開始され、図１２のステップ３１０でのファイル１０００（図９参照）から読み込まれた情報に基づいて進行する。ステップ１７２０でメモリが補助記憶に配置される。補助記憶の目的は、非持続フレームの表示の後で復帰されなければならないスクリーンの区域を格納することであり、それはアニメーション命令の持続フィールドが「偽」の値を有するときである。補助記憶のサイズは復帰に要求される最大の区域を決定するために、命令の構文解析を行うことで計算可能である。一方他の方法としては、補助記憶はアニメーションを表示するために使用されるスクリーン区域と同一のサイズで生成されてもよい。この後者の例では命令解析は不要である。

【００６３】アニメーションが持続フレームのみを含む場合、補助記憶は不要であり、ステップ１７２０は無効である。補助記憶のサイズに関連する情報は、ファイルヘッダの一部として格納されることを留意されたい。ステップ１７３０でメモリがレンダリング画像のために配置される。レンダリング画像の目的は、スクリーンメモリにコピーされるに先立ってフレームが合成される、画面外作業領域として働くことである。本実施形態においては、レンダリング画像は、アニメーションを表示するために使用されるスクリーン区域と同一のサイズである。実際には、レンダリング画像は小さいことも可能であるが、通常は、最低どの瞬間においてでも更新される最大のスクリーン区域と同一のサイズであり、この場合は命令の実行の結果得られる「レンダリング後の長方形」の最大のサイズである。他方で、一度にスクリーンにレンダリングされるべき最大区域（単一命令、またはフレーム定義シーケンスの最終命令の実行の結果として得られる）のサイズと少なくとも等しいと考えることができる。

【００６４】ファイルの第１のレイヤが完全に不透明であり、画像表示区域（ヘッダの幅および高さフィールドによって特定される）の全体を覆う場合には、背景記憶装置（ｂａｃｋｇｒｏｕｎｄ　ｓｔｏｒｅ）の配置は必要とされないことに注意されたい。また、補助記憶は、アニメーション制御ブロック中の命令のすべてが「真」の持続の値を有する場合、補助記憶は必要とされない。ステップ１７４０において、メモリはレイヤメモリに配置される。レイヤメモリの目的は２つあり、すでに復号されレンダリングされたが、後続のフレームにおいて再使用される画像データのためのキャッシュを提供すること、および再使用されることになっており、そこに含まれているレイヤを追跡する機構を提供することである。

【００６５】これらの目的を達成するために、レイヤメモリの各エントリは、そのレイヤのための画像データが検索されるためのハンドル、およびレイヤの再使用の前に実行されるべき命令の番号を記録する「次」とラベルされた変数を備えている。

【００６６】最後にステップ１７５０で、アニメーション制御ブロック中の「ループ」フィールドの値が零であるかを決定する判定が実行される。これが偽（すなわち、ステップ１７５０が「ｎｏ」と判断する）であれば、全アニメーションシーケンス３７２（図１２参照）が繰り返される。これをサポートするために、ステップ１７６０で付加的な背景記憶装置が配置され、初期スクリーン背景がステップ１７７０で背景記憶装置にコピーされる。「ループ」の値が零（すなわちステップ１７５０が「ｙｅｓ」と判断する）であれば、背景記憶装置は不要であり、実行は直接ステップ１７８０へ抜ける。

【００６７】図１２のステップ３２０は、図１５を参照して詳説される。実行は１８００で開始され、ステップ１８０２で変数が初期化される。特に、「復帰」は「偽」にセットされる。この変数は、補助記憶から背景が復帰される必要があるときを示す。「第１フレーム」の値は真にセットされ、これはアニメーションシーケンスの第１フレームが処理されようとしていることを示す。「タイマ」は、現在時間で初期化される。この変数は、アニメーションシーケンスの独立したフレームがスクリーンに現れる時間を決定するために使用される。最終的に、「レンダリングされた長方形」と、ラベルされた変数は、４つの零を含むように初期化される。レンダリングされた長方形は、スクリーン上の表示に関連して変化するレンダリングされた画像中の領域の原点（ｘおよびｙ）、およびサイズ（幅および高さ）を含む。これはスクリーンの更新の間使用される。

【００６８】ステップ１８０４において、レイヤメモリの各項目は参照され、項目「次」フィールドは値零にリセットされる。この値は、関連する画像ハンドルを解放することを保証する意味である。ステップ１８０６で背景記憶装置が使用されているか、を決定する判定が行われる。それはアニメーションシーケンス３７２（図１２参照）が少なくとも１回はループすることを示すものである。ステップ１８０６で「ｎｏ」と判断されれば、スクリーン画像は単に、ステップ１８０８で補助記憶にコピーされる。ステップ１８０６で「ｙｅｓ」と判断されれば、スクリーンの背景は、前のループ実行からの出力で損なわれている可能性があるので、背景記憶装置中に含まれる背景のコピーが使用されなければならない。こ

れはステップ１８１２で補助記憶へコピーされる。そして、実行はステップ１８１０へ抜ける。

【００６９】図１２のステップ３２５は図１６を参照して詳説される。実行はステップ１９００で開始する。後続のステップ１９０５で、「第１フレーム」変数の値が真であるかを決定する判定が実行される。これは、アニメーションプロセス３７２（図１２参照）がアニメーションシーケンスの最初にあることを示す。ステップ１９０５で「ｙｅｓ」と判断されれば、「現在のセット」変数は、ステップ１９１０で、アニメーション制御ブロック中で定義されている、第１の命令のセットを指すようにセットされ、後続のステップ１９１５で、「カウント」変数は、前述の現在のセット中の繰り返しフィールドの値に初期化される。ステップ１９６０で、変数「命令」は、実行がステップ１９６５で抜ける前に、現在のセット中の第１の命令を指すようにセットされる。

【００７０】ステップ１９０５で「ｎｏ」と判断され、第１のフレームに続くフレームがアニメーション化されることを示せば、どの命令を使用するかを決定するために、いくつかの付加的な判定が要求される。ステップ１９２０で、「命令」がすでに現在のセット中の最後の命令を指しているかどうか、を決定する判定が実行される。ステップ１９２０で「ｎｏ」と判断され、現在のセットがまだ終了していないことを示せば、ステップ１９４０で、「命令」が増分され、順序通りに現在のセット中の次の命令を指すようにされ、その後ステップ１９６５へと抜ける。

【００７１】ステップ１９２０で「ｙｅｓ」と判断され、現在のセット中の最後の命令が実行されてしまったことを示すならば、カウント変数が、ステップ１９２５で判定され、カウント変数が零であるかどうかが決定される。ステップ１９２５で「ｎｏ」と判断され、このセット中の命令が繰り返されなければならないことを示すならば、ステップ１９４５で「カウント」の値は、減分されて、「命令」はステップ１９６０で現在のセット中の第１の命令を指すようになされ、続いて実行はステップ１９６５へ抜ける。

【００７２】ステップ１９２５で「ｙｅｓ」と判断され、この命令のどの繰り返しも完了していること、および実行は次の命令セットの第１の命令で続行すべきである、ということを示すならば、ステップ１９３０で、現在のセットが、アニメーション制御ブロック中で定義されている最後の命令セットであるかどうか、を決定する判定が実行される。ステップ１９３０で「ｙｅｓ」と判断され、現在のセットが最後のセットであることを示すならば、変数「命令」は、どの命令ももはや使用可能状態にないことを示す特別な既定値にセットされる。図１６ではこの値は「停止」と表されている。ステップ１９３０が「ｎｏ」と判断し、アニメーション制御ブロックで定義され、さらに進められるべき命令セットがあることを示すならば、変数「現在のセット」はステップ１９５０で順序で従って次の命令セットを指すようにセットされ、変数「カウント」は、ステップ１９５５でその命令セットのための「繰り返し」フィールドの値に初期化される。続いて、変数「命令」はステップ１９６０で、新たな現在のセット中の第１の命令を指すようにセットされ、ついで実行はステップ１９６５へ抜ける。

【００７３】図１２のステップ３３５は図１７を参照して詳説される。実行はステップ２０００で開始される。続くステップ２００５で、「現在のレイヤ」とラベルされた変数は、「空」と示されている特別の値に初期化され、「Ｎ」とラベルされた変数は、レイヤメモリのエントリの数に初期化される。この変数（Ｎ）は、レイヤメモリ中の各エントリを処理するために、後続のループ命令中で使用される。ループ実行は続くステップ２０１０で開始され、「Ｎ」の値が零であるかを決定する判定が実行される。ステップ２０１０が「ｙｅｓ」と判断するならば、ループは抜けて、プロセス３３５はステップ２０１５へ向けられ、そこにおいて、「現在のレイヤ」の値が、特別な値「空」以外の何物かにセットされているかどうかを決定するための判定が実行される。ステップ２０１５が「ｙｅｓ」と判断するならば、現在のレイヤに関連するピクセルデータは、続くステップ２０２０で検索される。どちらのケースにおいても、実行はステップ２０６０へと抜ける。

【００７４】ステップ２０１０が「ｎｏ」と判断し、レイヤメモリのエントリはすべて参照されているわけではないことを示すならば、ループの主要な部分が実行される。ステップ２０２５で、変数ＬはレイヤメモリのN番目のエントリを指すようにセットされる。続くステップ２０３０で、そのエントリの「次」フィールドの値が零であるかどうかを見る判定が行われる。ステップ２０３０が「ｙｅｓ」を判断するならば、レイヤはステップ２０３５でレイヤメモリから除去される。これは整理ステップである。ステップ２０３０が「ｎｏ」を判断するならば、続くステップ２０４０で、エントリ「Ｌ」の「次」フィールドの値が、１に等しいかどうか、および現在のレイヤの値が「空」で表される特別な値に等しいかどうかを決定するために、判定が実行される。

【００７５】ステップ２０４０が「ｙｅｓ」を判断するならば、現在のレイヤはエントリ「Ｌ」中に含まれるレイヤにセットされ、そのエントリの「次」フィールドは現在の命令の「次」フィールドの値にセットされる。次いで、値「Ｎ」は、ステップ２０５５で減分され、プロセス３３５の実行はステップ２０１０へループバックする。ステップ２０４０が「ｎｏ」を判断するならば、エントリ「Ｌ」の次のフィールドの値はステップ２０５０で減分され、次いでステップ２０５５で「Ｎ」が減分されて、ステップ２０１０へループバックする。

【００７６】図１２のステップ３４５は、図１８（ａ）

および図１８（ｂ）を参照して詳説される。実行は図１８（ａ）のステップ９００から開始される。続くステップ９０５は、寿命が零の非持続フレームの特殊なケースを判定する。ステップ９０５が「ｙｅｓ」と判断し、この条件が存在していることを示すならば、プロセス３４５の実行は、即時ステップ９９５（図１８（ｂ）参照）へ抜ける。ステップ９０５が「ｎｏ」を判断すれば、レンダリングされた長方形の値は続くステップ９１０で更新され、レンダリングされた長方形の現在の値、および現在の命令で定義されたスクリーンの長方形の組を含むようになされる。続くステップ９１５で、切り取り操作が現在の命令によって要求されているかどうかを決定する判定が実行される。ステップ９１５が「ｙｅｓ」を判断し、切り取り操作が要求されていることを示しているならば、１つの実施形態においては、現在の命令の１つの範囲でのみ、現在のレイヤは、ステップ９２０で切り取られた領域と置き換えられる。どちらのケースにおいても、実行は次のステップ９２５へ移動する。

【００７７】ステップ９２５では、現在の命令によってリスケーリングが要求されているかどうかを決定するための判定が実行される。ステップ９２５が「ｙｅｓ」と判断し、リスケーリングが要求されていることを示せば、現在の命令の範囲でのみ、現在のレイヤは、ステップ９３０で、現在の命令で定義されているように、幅ｗ＿スクリーンおよび高さｈ＿スクリーンに切り取られた現在のレイヤの版と、置き換えられる。どちらのケースにおいても、プロセス３４５の実行はステップ９３５へ移動し、そこにおいて、現在の命令によって特定される位置（ｘ＿スクリーン、ｙ＿スクリーン）で、現在のレイヤは、レンダリング画像に重ねて、現在の画像の左上と合成される。ステップ９２０、９３０、９３５は、現在の実施形態の領域において、より最適化された様式で結合されることができる。実際に、これらの作業の１つまたは複数を結合した最適化された操作が使用される。本説明で、独立しており曖昧さを排除した処理ステップに分解して説明を行うのは、純粋に明確さを求めるという理由からである。

【００７８】続くステップ９４０（図１８（ｂ）参照）では、レイヤが持続し、タイマ刻みの零より大きい寿命を有しているかどうかを決定するための判定が実行される。ステップ９４０が「ｙｅｓ」値を判断すれば、このことは、レンダリング画像が、現在の表示と組み合わせて次のフレームを定義するための十分な情報を含んでいることを示しており、次いで実行は判定ステップ９５５へ移行し、そこで変数「復帰」の値が判定される。値「真」が判断されれば、プロセス３４５は「ｙｅｓ」矢印に従ってステップ９６５へ導かれ、そこにおいてレンダリングされた長方形によって特定される領域は、レンダリング画像からスクリーン画像へコピーされる。判定ステップ９５５が「偽」値を判断すれば、プロセス３４５は「ｎｏ」矢印に従ってステップ９６０へ導かれ、そこにおいて現在のスクリーンは補助記憶へコピーされ、プロセス３４５はその後ステップ９６５へ導かれる。

【００７９】後続のステップ９６５では、プロセス３４５はステップ９７０へ導かれ、そこにおいて、レンダリングされた長方形によって特定される領域は、補助記憶からレンダリング画像へコピーされる、その後にステップ９７５でレンダリングされた長方形は（０，０，０，０）へセットされ、変数「復帰」は「真」にセットされる。プロセス３４５は、その後ステップ９８０へ導かれ、そこでプロセス３４５は「現在時刻」がタイマの値よりも大きくなるまで待たされる。

【００８０】ステップ９４０では、値「偽」が判断すれば、プロセス３４５は、「ｎｏ」の矢印に従って、ステップ９４５へ導かれ、そこでレンダリングされた長方形によって特定される領域はレンダリング画像からスクリーン画像へコピーされる。その後ステップ９５０で「復帰」の値は「偽」にセットされ、プロセス３４５はステップ９８０へ導かれる。

【００８１】ステップ９８０の後に、プロセス３４５はステップ９８５へ導かれ、そこでレンダリングされた長方形によって特定されるスクリーン区域が更新される。その後ステップ９９０においてタイマが「現在時刻」に「寿命」を加えた値に等しくセットされ、次いでプロセス３４５はステップ９９５で終了する。

【００８２】図１２のステップ３６７は図１９を参照して詳説される。ステップ１３００で開始された後、プロセス３６７はステップ１３１０へ導かれ、そこでレンダリングされた長方形によって特定される領域はレンダリング画像からスクリーン画像へコピーされる。その後ステップ１３２０で、レンダリングされた長方形によって特定されるスクリーン区域が更新され、次いでプロセス３６７はステップ１３３０で終了する。

【００８３】あるいはアニメーションを提供する方法は、アニメーションを提供する機能またはサブ機能を実行する、１つまたは複数の集積回路のような、専用のハードウェアで実現されてもよい。このような専用のハードウェアは、グラフィックプロセッサ、ディジタル信号プロセッサ、または１つまたは複数のマイクロプロセッサおよびそれに連動するメモリを含んでいてもよい。

【００８４】図２０は、第１の画像が第２の背景画像へ「滑り込ませる」アニメーションシーケンスに関連する命令セットの例である。この図は、９列の整数１２０８～１２２４および２つのファイル指示１２２６および１２２８を含む命令セット１２００を示している。最初の３列１２０８～１２１２の最上部の整数１２０２～１２０６は背景画像のそれぞれ幅および高さに、およびアニメーション中で使用されるレイヤ（すなわち画像）の数に関連するヘッダ情報を提供する。９つの列１２０８～１２２４（最初の３列を除いて）は、それぞれ、ｘ＿ス

クリーン、ｙ＿スクリーン、ｘ＿切り取り、ｙ＿切り取り、ｗ＿切り取り、ｈ＿切り取り、「寿命」、「持続」、「次」の変数を表す。ヘッダ情報は別として、列は１１行を含み、アニメーションは１１ステップで実行されることを示す。各行は命令を表し、この１１行は単一の命令セットを表す。

【００８５】図２１は図２０のアニメーションシーケンスを表す。図では、ブランクの背景上の円盤（すなわち図２０のファイル１２２８）の背景画像１４０８を示している。画像１４０８の寸法は、６７５（すなわち図２０の１２０２）の幅（矢印１４１０で示す）で、４５０（すなわち図２０の１２０４）の高さ（矢印１４０６で示す）である。図ではまた、背景画像１４０８に滑り込ませるべき画像１４０４（すなわち図２０の１２２６）を示している。この画像１４０４は、それぞれ矢印１４００および１４０２で示される、幅および高さを有する。背景画像の画像１４０４が、連続的に背景画像上へ滑り込む後続の４つの展望図が示されている。

【００８６】命令セット１２００の第１の行１２３０（図２０参照）は、背景画像１４１２を配置する。ｘ＿スクリーンおよびｙ＿スクリーンの値である行１２３０の左から最初の２つの数字は、その画像が表示区域の左上コーナに、その画像の左上コーナを重ねて配置されるべきであることを示している。この行１２３０の「次」の値、すなわち最右端の整数は値が「零」であるので、これは現在の画像またはレイヤは再使用されないことを示し、後続の画像、この場合は画像１４０４、が処理する次のものであることを示している。

【００８７】続く行１２３２は、その結果画像１４０４を処理する。ｘ＿スクリーンおよびｙ＿スクリーンの値である、行１２３２の左から２つの整数は、画像１４０４が表示区域の左上コーナにその左上コーナを重ねて配置されるべきであることを示している。行の左から３番目および５番目の整数は、すなわちｘ＿切り取りおよびｗ＿切り取りは、現在の命令によりｘ方向に「持続」されるべき画像１４０４のその部分を示す。これは、左より画像１４０４に沿ってｘ＿切り取り（すなわち４００）の分だけ移動すること、および次の画像のｗ＿切り取り（すなわち４５）の分だけを持続させることによって実行される。同様に、列の左から４番目および６番目の整数は、すなわちｙ＿切り取りおよびｈ＿切り取りは、現在の命令によりｙ方向に「持続」されるべき画像１４０４のその部分を示す。これは、最上部より画像１４０４に沿ってｙ＿切り取り（すなわち０００）の分だけ下ること、および次のｈ＿切り取りの分、すなわち１２４だけ持続させることによって実行される。このケースの場合は画像全体を表す。したがって、右「４５」、全高「１２４」の画像が持続されるべきであり、これがｘ＿スクリーン、ｙ＿スクリーン、すなわち左上の原点から（０，０）のずれの位置に配置される。この結果が図２０に１４１４で示されており、画像１４０４が背景画像上に部分的に滑り込んでいることを示している。さらに行１２３２の左から７番目の整数、すなわち「寿命」は値１を有することを考慮すれば、これは現在の命令の実行の完了および次の命令の実行の間で、１刻みが発生することを示している。「寿命」の値は、画像１４０４がなめらかに滑り込むという結果を生じる。

【００８８】さらに行１２３２について考察すると、左から８番目の整数、すなわち「持続」は値零を有し、次の命令の実行に先立ち、スクリーン値が背景の実行前にリセットされることを意味する。

【００８９】命令行１２３２の最右列は「次」の値を１にし、それは現在のレイヤ（画像１４０４）が次の命令で使用されることを意味し、そこにおいて背景の左上コーナに重ねて、少し大きい区域が切り取られ、レンダリングされる。

【００９０】１４１６および１４１８には、画像１４０４が画像１４０８上に滑り込みつつある進行状態が示されている。

【００９１】［産業上の利用分野］本発明の実施形態が、上述のように、コンピュータデータ処理産業に対して応用可能であり、特にこれらの産業の区分に対して応用可能であることは明らかである。さらに、本発明の実施形態はまた、広告および娯楽産業に対して応用可能である。

【００９２】前述した内容は、本発明のいくつかの実施形態を記述したに過ぎず、改良および／または変更は、実例となり制限となることのない、本発明、実施形態の領域および趣旨から逸脱することなく可能である。

【図面の簡単な説明】

【図１】本発明の実施形態が実行できる汎用コンピュータの概略ブロック図である。

【図２】本発明の実施形態に従うファイルフォーマットの例を示す図である。

【図３】本発明の実施形態に従うファイルフォーマットの別の例を示す図である。

【図４】本発明の実施形態で使用される座標配列を示す図である。

【図５】実施形態のループ処理機構のための制御の概略の流れ図である。

【図６】図５の「現在のレイヤのための制御ブロックを読む」ステップをより詳細に示す流れ図である。

【図７】図５の「レイヤを表示する」ステップをより詳細に示す流れ図である。

【図８】図５の「現在のレイヤの値の更新」ステップをより詳細に示す流れ図である。

【図９】本発明の実施形態に従う画像ファイル構造を示す図である。

【図１０】レイヤの区画がレンダリングされるべき仮想スクリーンを示す図である。

【図１１】本発明の実施形態に従う、アニメーション化された画像シーケンスを支援するメモリ配置を示す図である。

【図１２】本発明の実施形態に従うアニメーションプロセスを示す方法のステップの流れ図である。

【図１３】図１２中のヘッダおよびアニメーション制御ブロック情報を読み込むステップに関連する方法のステップの流れ図である。

【図１４】図１２中のスクリーンメモリおよびサポート構造の配置のステップに関連する方法のステップの流れ図である。

【図１５】図１２中のメモリおよびサポート構造の初期化のステップに関連する方法のステップの流れ図である。

【図１６】図１２中の命令決定のステップに関連する方＊法のステップの流れ図である。

【図１７】図１２中の画像レイヤ決定のステップに関連する方法のステップの流れ図である。

【図１８（ａ）】図１２中の画像レイヤのレンダリングのステップに関連する方法のステップの流れ図である。

【図１８（ｂ）】図１２中の画像レイヤのレンダリングのステップに関連する方法のステップの流れ図である。

【図１９】図１２中のレンダリングされた長方形の流し込みのステップに関連する方法のステップの流れ図である。

【図２０】実施形態に従っている画像ファイル構造をアニメーションシーケンスと共に例示する図である。

【図２１】図２０のアニメーションシーケンスを示す図である。

【図１】

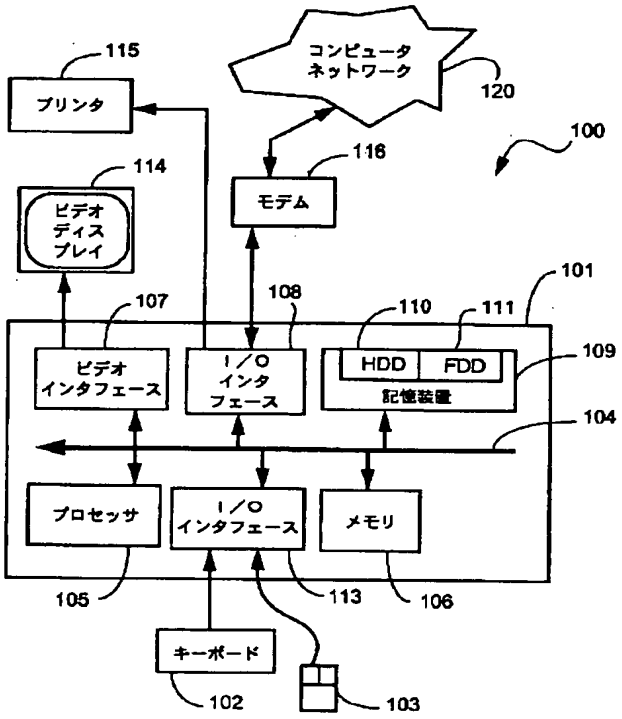


Fig. 1

【図２】

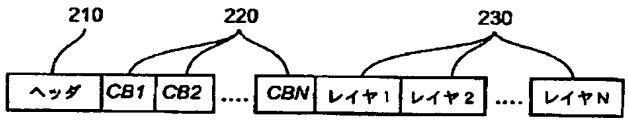


Fig. 2

【図３】

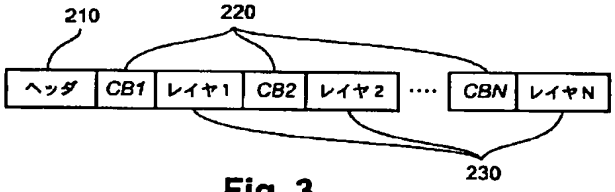


Fig. 3

【図４】

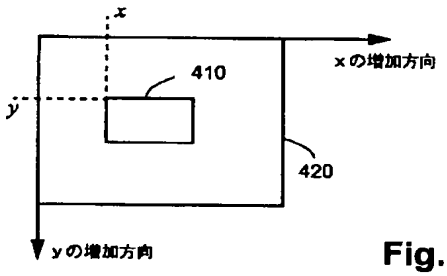


Fig. 4

【図５】

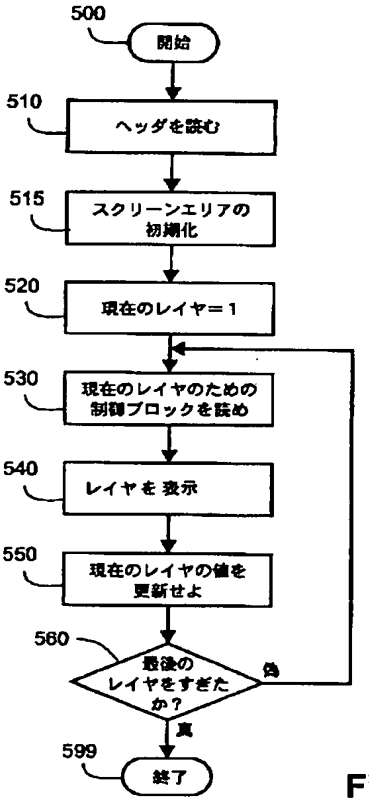


Fig. 5

【図６】

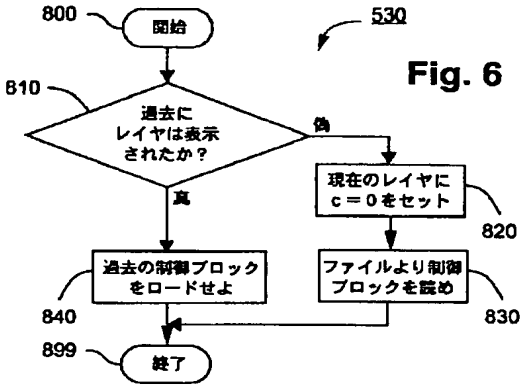


Fig. 6

【図１０】

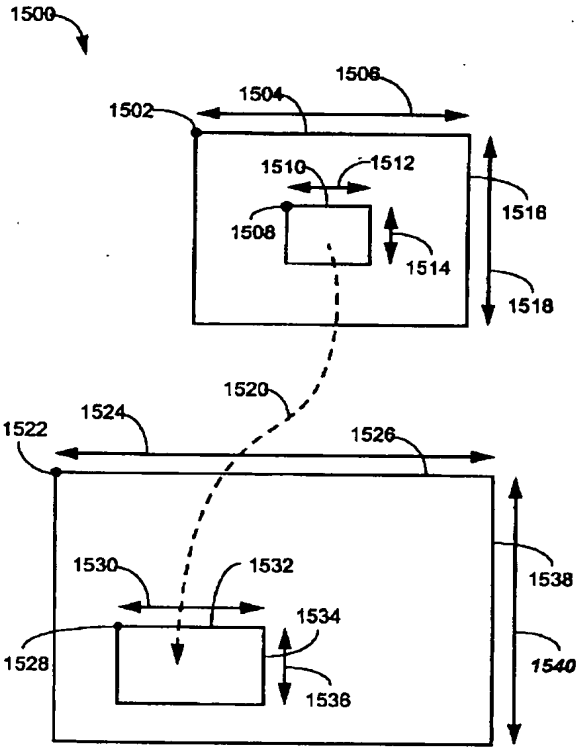


Fig. 10

【図７】

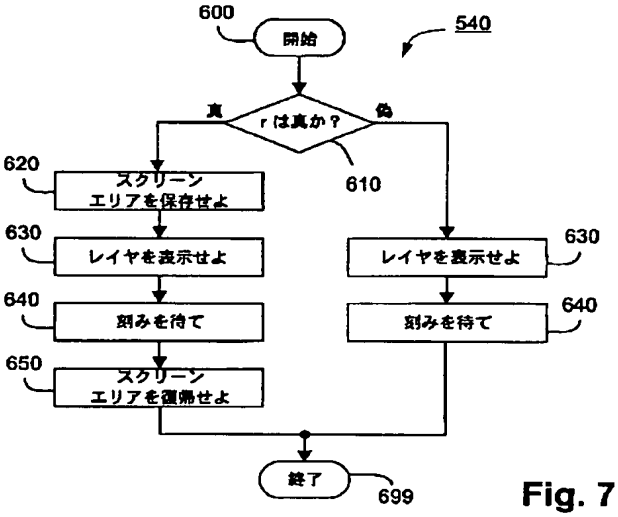


Fig. 7

【図８】

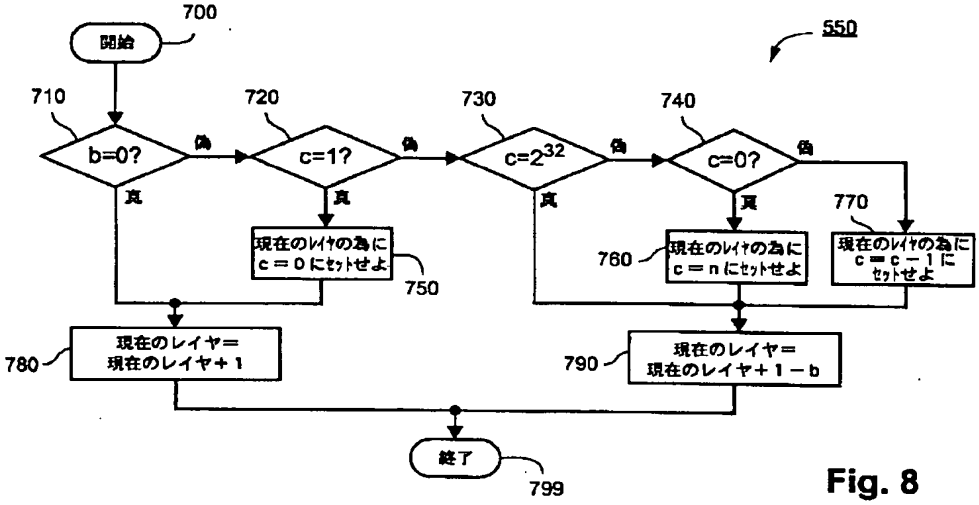


Fig. 8

【図９】

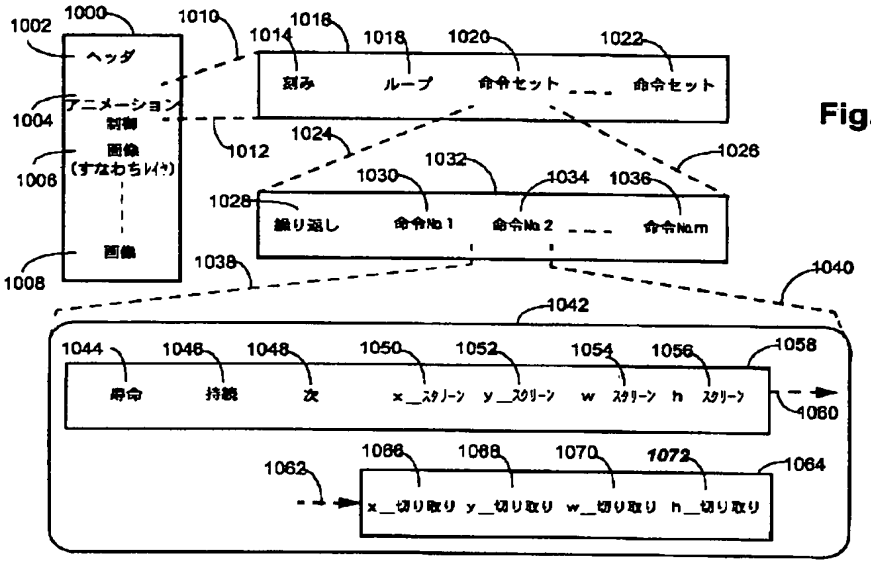


Fig. 9

【図１１】

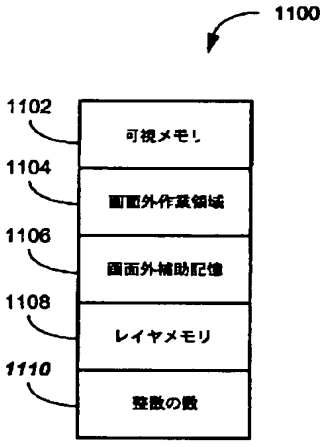


Fig. 11

【図１２】

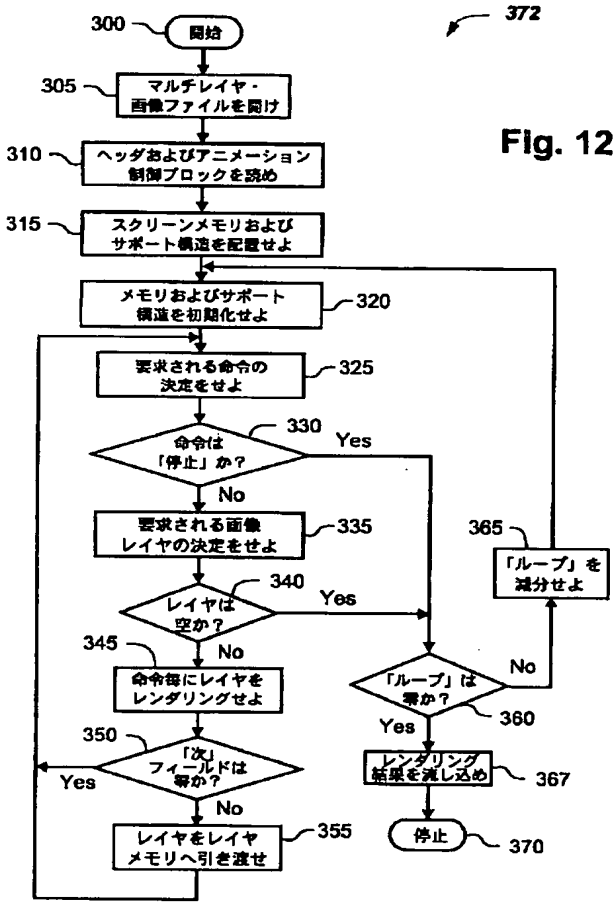


Fig. 12

【図１３】

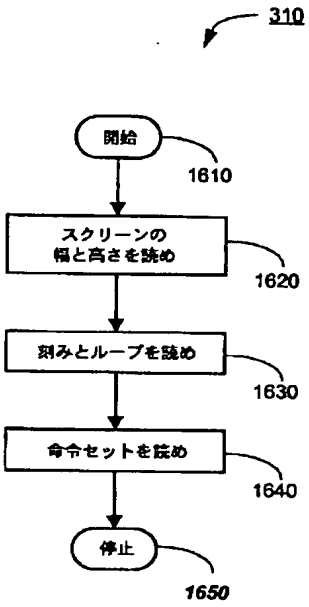


Fig. 13

【図１７】

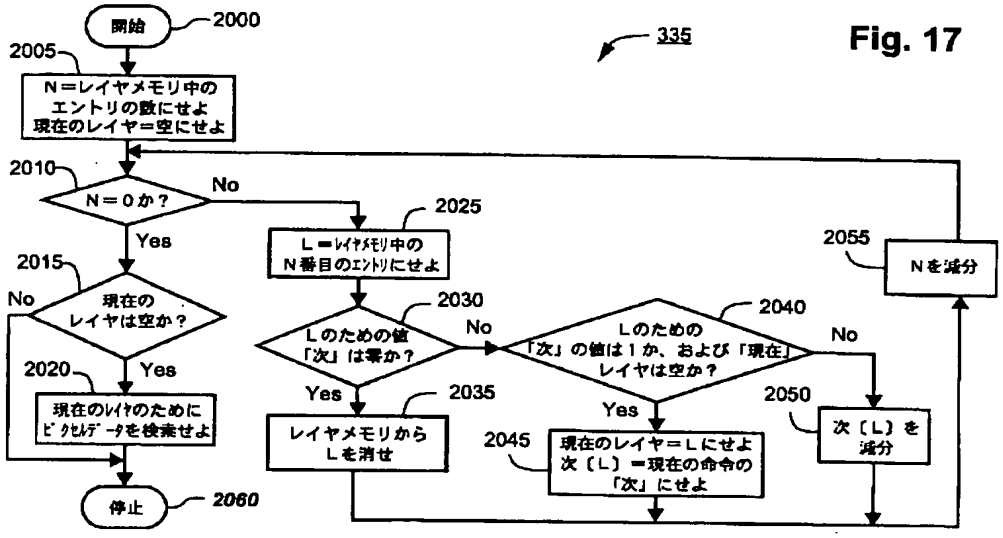


Fig. 17

【図１４】

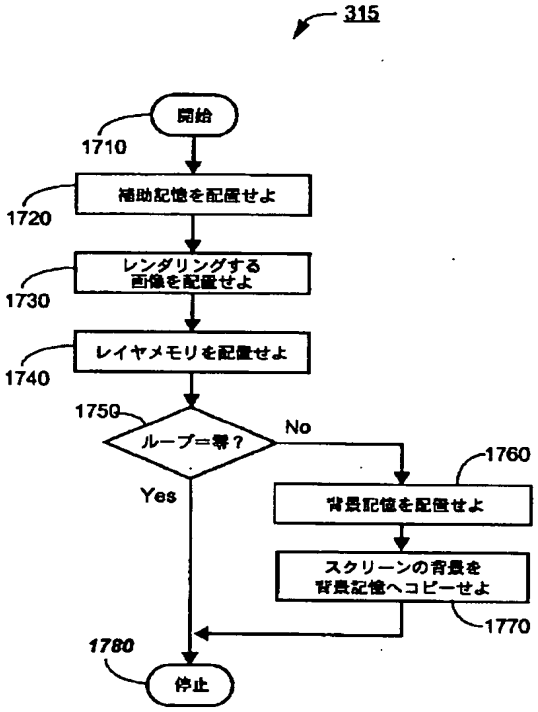


Fig. 14

【図１５】

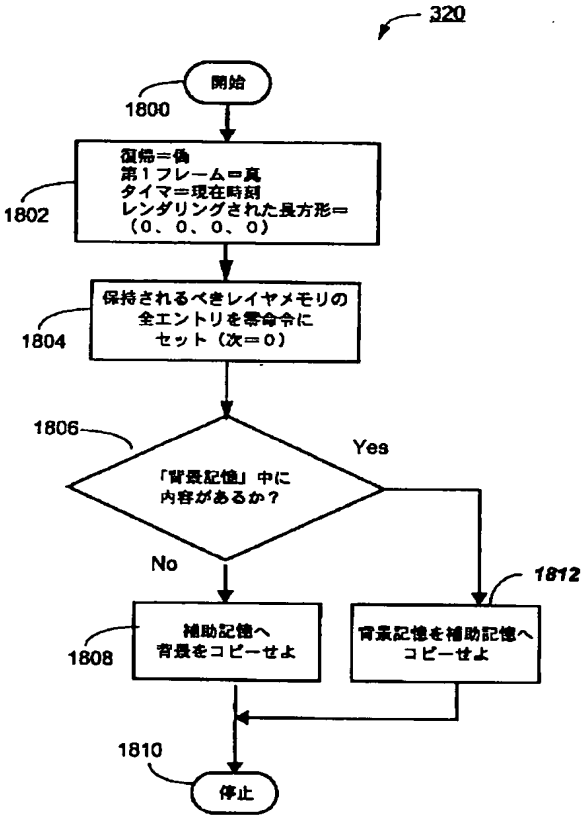


Fig. 15

【図１６】

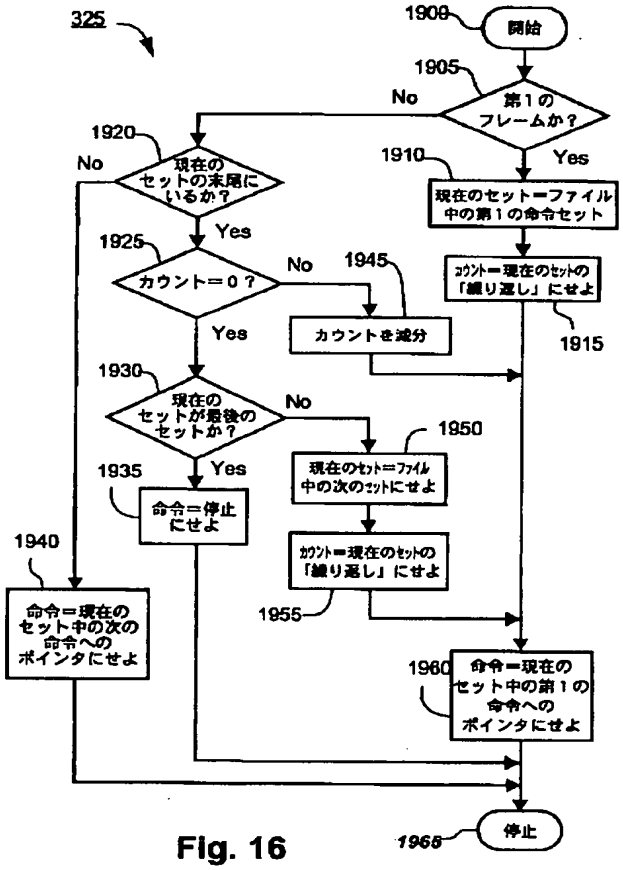


Fig. 16

【図１８（ａ）】

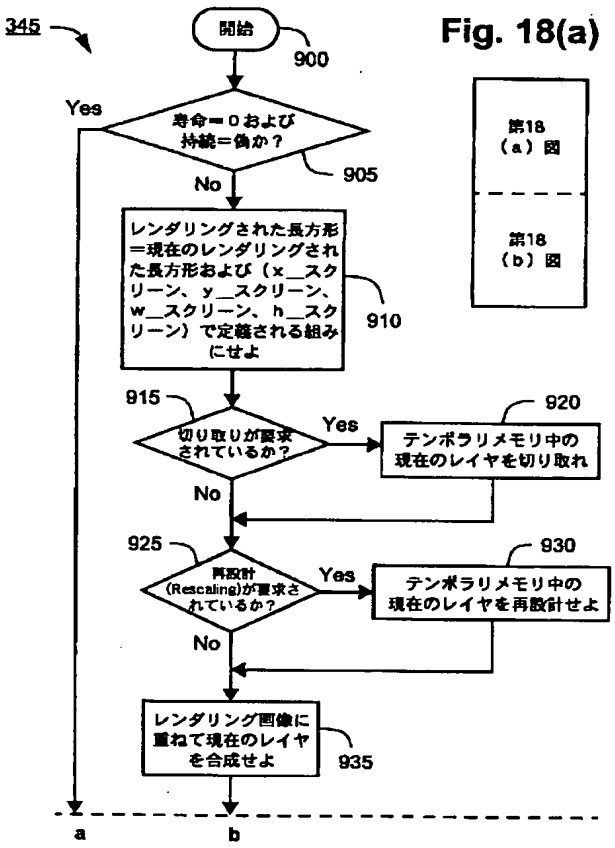


Fig. 18(a)

【図１８（ｂ）】

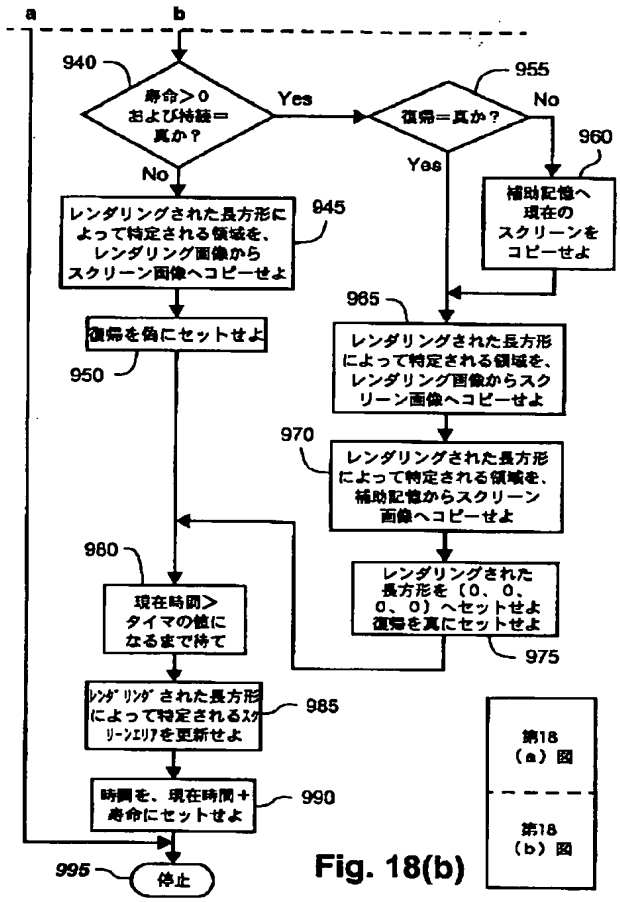


Fig. 18(b)

【図１９】

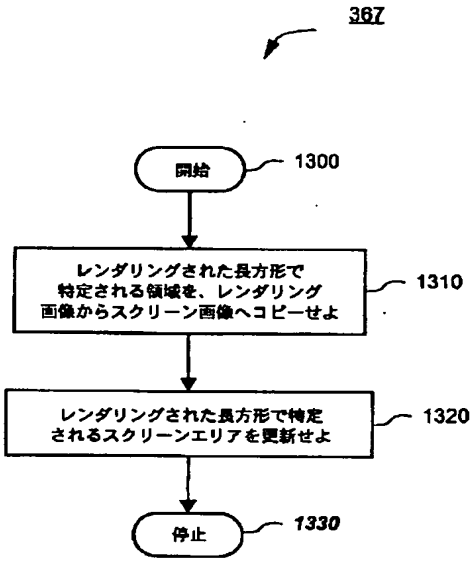


Fig. 19

【図２０】

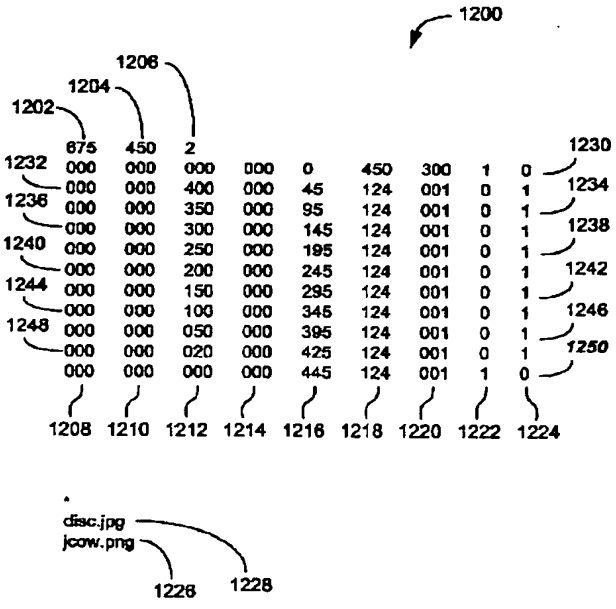
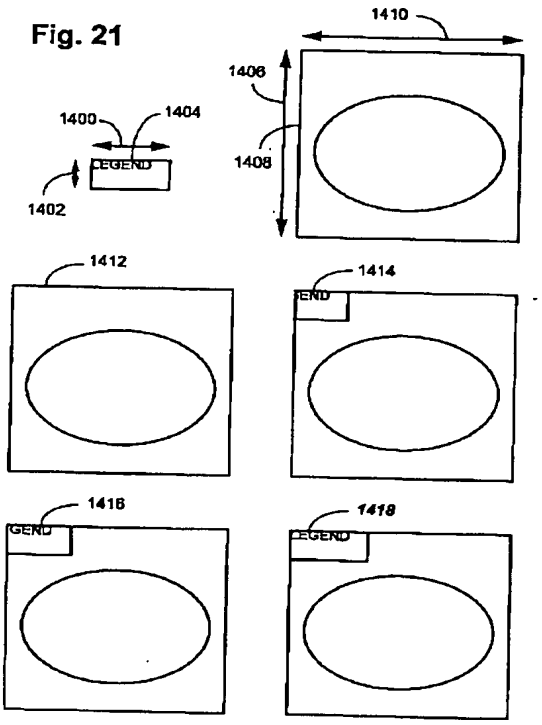


Fig. 20

【図２１】

Fig. 21



【外国語明細書】

1. Title of the Invention

A METHOD FOR ENCODING ANIMATION IN AN IMAGE FILE

2. What is claimed is

1. A method of processing a multi-layer image file comprising (i) a first plurality of image layers, and (ii) a second plurality of control blocks, said processing producing an animation sequence, said method comprising steps of:

providing a plurality of control blocks in said image file, each control block being associated with at least one of said image layers, wherein each control block is characterised by an information control field indicating which one of said control blocks and associated image layers to loop back to; and

sequentially executing each control block and looping back to a previous control block and associated layer in said execution sequence in accordance with the indication provided by said information control field.

2. A method according to claim 1, wherein said control block further includes a loop field number value which indicates how many times to loop back to the layer indicated by the information control field.

3. A method of animating a sequence of images, wherein said images are contained in a single multi-layer file, the method comprising the steps of:

a) providing a plurality of control blocks in said image file, each control block being associated with at least one of said images, wherein each control block is characterized by an information control field indicating which

one of said images is next in sequence, and at least one control block having an information control field indicating a previous image in the sequence;

b) reading a current information control field from a current control block;

c) displaying the image associated with the current control block;

d) if the current information control field indicates a loop-back to a previous image then taking the control block of the previous image as the current control block, otherwise taking the control block of a next image in the sequence as the current control block; and

e) repeating steps b) through to e).

4. A method for providing an animation of one or more images of a plurality of images, said method comprising steps of:

storing said plurality of images in a first order;

determining a commencing image of said animation;

determining a commencing address for the commencing image;

establishing an animation order for said one or more images, said animation order commencing with said commencing image;

animating said one or more images dependent upon said animation order, using relative addressing referred to said commencing address; and

reusing at least one image of said one or more images, if said at least one image occurs more than once in the animation order.

5. A method of processing a multi-layer image file comprising (i) a first plurality of image layers, and (ii) a second plurality of control blocks, said

processing producing an animation sequence, said method comprising steps of:

processing an image layer in accordance with a corresponding control block, thereby providing an image for said animation sequence;

tagging the image layer for reprocessing, if the image layer is to be used again in the image sequence, said tagging using a relative address referred to an address of said corresponding control block;

re-using said image layer in accordance with a next control block, thereby providing a next image for the animation sequence, if said relative address is a next address to the address of said corresponding control block; and

using a next image layer in accordance with said next control block, thereby providing said next image for the animation sequence, if said relative address is a subsequent address to said next address.

6. A method of processing a multi-layer image file comprising (i) a first plurality of image layers, and (ii) a second plurality of control blocks, said processing producing an animation sequence, said method comprising steps of:

processing an image layer in accordance with a corresponding control block, thereby providing an image for said animation sequence;

tagging the image layer for reprocessing, if the image layer is to be used again in the image sequence, said tagging using a relative address referred to an address of said corresponding control block;

re-using said image layer in accordance with a later control block,

thereby providing a later image for the animation sequence, if said relative address is a later address referred to the address of said corresponding control block; and

using a next image layer in accordance with a next control block, thereby providing a next image for the animation sequence, if said relative address is a subsequent address to said next address.

7. A method according to claim 5, wherein said processing, tagging, re-using and using steps are repeatedly performed a first number of times in a first loop, said first number being dependent upon a "repeat" parameter.

8. A method according to claim 7, wherein said first loop is repeatedly performed a second number of times in a second loop, said second number being dependent upon a "loop" parameter

9. A method according to claim 5, wherein a time interval between provision of said image, and provision of said next image for said animation sequence is determined substantially by a "life" parameter.

10. A method according to claim 5, wherein pixels of said image rendered to a screen persist on the screen dependent upon a "persist" parameter.

11. An apparatus for processing a multi-layer image file comprising (i) a first plurality of image layers, and (ii) a second plurality of control blocks, said processing producing an animation sequence, said apparatus comprising:

processing means for processing an image layer in accordance with a corresponding control block, thereby providing an image for said animation sequence;

tagging means for tagging the image layer for reprocessing, if the image layer is to be used again in the image sequence, said tagging using a relative address referred to an address of said corresponding control block; and

re-use means for re-using said image layer in accordance with a next control block, thereby providing a next image for the animation sequence, if said relative address is a next address to the address of said corresponding control block.

12. An apparatus adapted to animate one or more images of a plurality of images, said apparatus comprising:

a file structure means for storing said plurality of images in a first order;

an anchor address means for determining a commencing address of a commencing image of said one or more images;

an instruction set means for establishing an animation order for said one or more images using relative addressing referred to said commencing address;

an animation means for providing an animation of said one or more images in said animation order; and

image re-use means for re-using at least one image of said one or more images if said at least one image occurs more than once in said animation

order.

13. A multi-layer image file encoded for animation, said image file comprising:

a first plurality of image layers; and

a second plurality of control blocks; wherein an image layer is processed in accordance with a corresponding control block, thereby providing an image for said animation sequence; and wherein

the image layer is tagged for reprocessing, if the image layer is to be used again in the image sequence, said tagging using a relative address referred to an address of said corresponding control block.

14. A multi-layer image file comprising:

(a) a plurality of images stored in a first order;

(b) a first animation instruction for execution, said first animation instruction comprising:

(i) a commencing address of a commencing image of said plurality of images to be animated;

(ii) at least one animation attribute of said commencing image;

(c) at least one next animation instruction to be executed, said first animation instruction and said at least one next animation instruction being executed in a sequential order, each said at least one next animation instruction comprising:

(i) a relative address of a next image of said plurality of images to be animated, said relative address being referred to one of said

commencing address and a preceding relative address;

(ii) at least one animation attribute of said next image.

15. A multi-layer image according to claim 14, wherein said at least one animation attribute is one of:

a "life" value denoting a target time interval between completion of execution of a current instruction and completion of execution of a next instruction;

a "persist" value denoting whether pixels rendered to screen as a result of execution of the current instruction appear to persist on the display background or appear to be reset to a pre-execution background;

a "next" value denoting a number of instructions to execute before reusing a current image;

a "location" value denoting a location in which to place the current image, unscaled, within a display area;

a "size" value denoting a scale factor to be applied to the current image for placing within the display area; and

a "crop" value denoting a region to crop from an image being acted on;

16. A multi-layer image according to claim 15, wherein a value of zero for the "next value denotes that the current image is not reused.

17. A computer readable memory medium for storing a program for apparatus which processes a multi-layer image file comprising (i) a first plurality of image layers, and (ii) a second plurality of control blocks, said

processing producing an animation sequence, said program comprising:

code for a processing step for processing an image layer in accordance with a corresponding control block, thereby providing an image for said animation sequence; and

code for a tagging step for tagging the image layer for reprocessing, if the image layer is to be used again in the image sequence, said tagging using a relative address referred to an address of said corresponding control block.

18. A computer readable memory medium according to claim 17, said program further comprising:

code for a re-using step for re-using said image layer in accordance with a next control block, thereby providing a next image for the animation sequence, if said relative address is a next address to the address of said corresponding control block; and

code for a using step for using a next image layer in accordance with said next control block, thereby providing said next image for the animation sequence, if said relative address is a subsequent address to said next address.

19. A computer readable memory medium according to claim 17, said program further comprising:

code for a re-using step for re-using said image layer in accordance with a later control block, thereby providing a later image for the animation sequence, if said relative address is a later address referred to the address of said corresponding control block; and

code for a using step for using a next image layer in accordance with a next control block, thereby providing a next image for the animation sequence, if said relative address is a subsequent address to a next address.

3. Detailed Description of Invention

Field of Invention

The current invention relates to multi-layered image file formats and in particular multi-layer image files which are intended to be displayed as an animation, or alternatively, in the context of a multi-layer composite image.

Background Art

Multi-layer (or multi-page) images can be thought of as a set of images, all typically but not necessarily the same size, which are somehow combined for the purpose of displaying on an output display device 114. Thus multi-layer refers to multiple images in a single file. Each image in the file is referred to as a layer. There are currently two significant applications areas for which multi-layer images are used and they include:

- image editing and graphic design; and
- animation, especially animation in web pages on the Internet.

In the area of image editing and graphic design, multi-layer images allow a composition of complex scenes as different images are layered over each other. In this case it is usual for each layer to have an opacity (or alpha) channel associated with it. To display the various layers (images) on a display device 114 a first layer (typically a background image) is rendered and subsequent layer is then composited upon the first layer, for example, according to the following equations.

$$Ac = 1 \_ (1 - At)(1 \quad Ab) \qquad (1)$$

$$s = At=Ac \qquad (2)$$

$$t = (1 \_ At)Ab = Ac \qquad (3)$$

$$Rc = sRt + tRb \qquad (4)$$

$$Gc = sGt + tGb \qquad (5)$$

$$Bc = sBt + tBb \qquad (6)$$

In the above equations: the background image is specified in the RGBA (Red, Green, Blue and Alpha) colour space as (Rb, Gb, Bb Ab); a foreground (or top) image is specified in the RGBA colour space as (Rt, Gt, Bt, At); and the output or composite image is specified in the RGBA colour space as (Rc, Gc, Bc, Ac). Each subsequent (or new) layer is taken to be a foreground image until it is combined (composited) with a previous layer, wherein the combination is then taken to be a (new) background image. In this manner it is possible to combine a number of layers by sequential applications of equations (4-6) to each new layer in turn in order to form a final composite image. Other compositing operations are also possible however the one described herein-before with reference to equations (1) to (6) is the most commonly used.

The other noteworthy application of multi-layer images, noted above, is animation. For this purpose, currently, the most widely used file format is the Graphics Interchange Format (GIF). The GIF also contains layers (or multiple images) which are composited in sequence order. Each layer of a GIF file may be of different size and is positioned using offset coordinates in order to improve storage efficiency in cases where only small areas contain changes from one layer to the next. The GIF standard defines a virtual

screen upon which each layer is composited. It uses a control block structure to indicate how the layers in the file are to be displayed. Each layer of the file format is preceded by a control block which contains: information about the location of the top left corner in the virtual screen; information on how long the layer should be displayed before proceeding to the next layer in the file; and whether the layer should be removed prior to display of a next layer in the file. This (control block based) structure allows for particularly simple software implementation of the decoder. In fact very little additional coding is required to implement a GIF decoder capable of correctly displaying multi-layer animated GIF images.

The animation scheme employed by GIF has been adopted widely in a very short space of time. The primary reason for this is the simple and restricted design. These features make it easy for a large number of independent developers to implement file viewers capable of handling GIF animations. However the simplicity of GIF comes at the price of efficiency in coding. For example, as each layer in an animated GIF file corresponds to a single display frame, animation using sprites and overlays is not coded efficiently. This is because each frame must be present as a separate image layer. Images that are reused through the course of an animation must appear once in the file for each frame they appear in.

More recently, the Multiple Image Network Graphics (MNG) file format, which is still being developed, has attempted to address this problem. MNG defines an animation framework based on extensions to the Portable Network Graphics (PNG) file format. However, while MNG permits the reuse of layers, much of the simplicity that characterised the success of GIF is

lost. In addition, the methods used by MNG to describe the animation do not lead naturally to an implementation model. This makes the development of viewers for MNG animations notably more difficult to implement. To help address this problem the creators of MNG have proposed low complexity and very low complexity subsets of the full MNG standard. The problem with this however is that the low complexity subsets achieve little more functionality than GIF and have the same coding efficiency problems.

Summary of the Invention

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to a first aspect of the invention, there is provided a method of processing a multi-layer image file comprising (i) a first plurality of image layers, and (ii) a second plurality of control blocks, said processing producing an animation sequence, said method comprising steps of:

providing a plurality of control blocks in said image file, each control block being associated with at least one of said image layers, wherein each control block is characterised by an information control field indicating which one of said control blocks and associated image layers to loop back to; and

sequentially executing each control block and looping back to a previous control block and associated layer in said execution sequence in accordance with the indication provided by said information control field.

According to a another aspect of the invention, there is provided an apparatus for processing a multi-layer image file comprising (i) a first plurality of image layers, and (ii) a second plurality of control blocks, said processing producing an animation sequence, said apparatus comprising:

processing means for processing an image layer in accordance with a corresponding control block, thereby providing an image for said animation sequence;

tagging means for tagging the image layer for reprocessing, if the image layer is to be used again in the image sequence, said tagging using a relative address referred to an address of said corresponding control block; and

re-use means for re-using said image layer in accordance with a next control block, thereby providing a next image for the animation sequence, if said relative address is a next address to the address of said corresponding control block.

According to a another aspect of the invention, there is provided a multi-layer image file encoded for animation, said image file comprising:

a first plurality of image layers; and

a second plurality of control blocks; wherein an image layer is processed in accordance with a corresponding control block, thereby providing an image for said animation sequence; and wherein

the image layer is tagged for reprocessing, if the image layer is to be used again in the image sequence, said tagging using a relative address referred to an address of said corresponding control block.

According to a another aspect of the invention, there is provided a computer readable memory medium for storing a program for apparatus which processes a multi-layer image file comprising (i) a first plurality of image layers, and (ii) a second plurality of control blocks, said processing producing an animation sequence, said program comprising:

code for a processing step for processing an image layer in accordance with a corresponding control block, thereby providing an image for said animation sequence; and

code for a tagging step for tagging the image layer for reprocessing, if the image layer is to be used again in the image sequence, said tagging using a relative address referred to an address of said corresponding control block.

Detailed Description of the Preferred Embodiments

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

One embodiment of the invention can preferably be practiced using a general-purpose computer, such as the one shown in Fig. 1, wherein the processes of Figs. 2 to 8, and Figs. 12-19 may be implemented as software executing on the computer. In particular, the steps of the encoding, decoding methods are effected by instructions in the software that are carried out by the computer. The coding algorithm for providing signalling to of the structure of a code stream representing a digital image may also be implemented by instructions in software carried out by the computer. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer preferably effects an

advantageous apparatus for encoding digital images, decoding or signalling the structure coded representations of digital images in accordance with the embodiments of the invention.

The computer system 100 consists of the computer 101, a video display 114, and input devices 102, 103. In addition, the computer system 100 can have any of a number of other output devices 115 including line printers, laser printers, plotters, and other reproduction devices connected to the computer 101. The computer system 100 can be connected to one or more other computers using an appropriate communication channel via a modem 116, a computer network 120, or the like. The computer network may include a local area network (LAN), a wide area network (WAN), an Intranet, and/or the Internet.

The computer 101 itself consists of a central processing unit(s) (simply referred to as a processor hereinafter) 105, a memory 106 which may include random access memory (RAM) and read-only memory (ROM), an input/output (IO) interface 108, a video interface 107, and one or more storage devices generally represented by a block 109 in Fig. 1. The storage device(s) 109 can consist of one or more of the following: a floppy disc 111, a hard disc drive 110, a magneto-optical disc drive, CD-ROM, magnetic tape or any other of a number of non-volatile storage devices well known to those skilled in the art. Each of the components 105 to 113 is typically connected to one or more of the other devices via a bus 104 that in turn can consist of data, address, and control buses.

The video interface 107 is connected to the video display 114 and provides video signals from the computer 101 for display on the video display

114. User input to operate the computer 101 can be provided by one or more input devices. For example, an operator can use the keyboard 102 and/or a pointing device such as the mouse 103 to provide input to the computer 101.

The system 100 is simply provided for illustrative purposes and other configurations can be employed without departing from the scope and spirit of the invention. Exemplary computers on which embodiments can be practiced include IBM-PC/ATs or compatibles, one of the Macintosh (TM) family of PCs, Sun Sparcstation (TM), or the like. The foregoing are merely exemplary of the types of computers with which embodiments of the invention may be practiced. Typically, the processes of the embodiments, described hereinafter, are resident as software or a program recorded on a hard disk drive (generally depicted as block 110 in Fig. 1) as the computer readable medium, and read and controlled using the processor 105. Intermediate storage of the program and pixel data and any data fetched from the network may be accomplished using the semiconductor memory 106, possibly in concert with the hard disk drive 110.

In some instances, the program may be supplied to the user encoded on a CD-ROM or a floppy disk (both generally depicted by block 109), or alternatively could be read by the user from the network via a modem device connected to the computer, for example. Still further, the software can also be loaded into the computer system 100 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and

information recorded on websites and the like. The foregoing are merely exemplary of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

Embodiments of the coding method may alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub-functions of the encoding, decoding or signalling processes. Such dedicated hardware may include ASICs and associated on-chip memories.

First Embodiment

Two multi-layer image file formats will now be described with reference to Fig. 2 and Fig. 3 of the accompanying drawings. In each example depicted in Fig. 2 and Fig. 3, the start of a file comprises a collection of header information 210 which indicates to a decoder a size of an output display device 114 (see Fig. 1) area upon which layers are to be displayed. For each layer in the file there is an associated control block 220. Denoted in Fig. 2 and Fig. 3 by "CB" followed by a numeral N. The numeral N indicates a correspondence between the Nth control block and the Nth layer. For example, a control block associated with "layer 2" is denoted as "CB2". The control blocks may be collected together into the header region of the file as shown in Fig 2, in which case their sequence order corresponds to the sequence order of image layers contained in the file. Alternatively, as illustrated in Fig. 3, control blocks may precede each image layer in the file. Image layers may be encoded using standard or proprietary coding schemes or may simply be represented as a sequence of raw pixel values. For

example, each layer in the file format can individually be encoded using a Wavelet based encoding scheme or alternatively JPEG (Joint Picture Expert Group) compression can be used on each layer.

Table 1 shows a plurality of information fields, and their associated descriptions, used to implement an embodiment of the present invention.

Field	Description
x	pixel offset from left edge of an output device display area to the left edge of the corresponding layer
y	Pixel offset from top edge of the output device display area to the top edge of the corresponding layer
t	The amount of time the display is to be maintained after rendering of the current layer prior to displaying a next layer
r	Indicates whether or not the current layer is to be removed (and the display restored to its previous state) prior to displaying a next layer.
b	Indicates, next layer to render conditional on n
n	Indicates the number of times to follow the branch defined by *b* prior to displaying the layer in the file, treating the sequentially next layer in the file as the next layer to render.

Table 1

Each control block contains a plurality of information fields, each field provides information regarding a layer corresponding to a control block. Referring to table 1, there is outlined a plurality of information fields according to an embodiment of the present invention and a corresponding description for each of the fields shown. Each field is represented in the file using a 32 bit unsigned integer stored so that the most significant byte appears first. Other representations could be used, including variable length representations, without departing from the spirit of the invention.

The first two fields of table 1, *x* and *y* are further described with reference to Fig. 4. They indicate a offset from the top left corner of the screen area 420 used to display the image of the top left corner of the corresponding image layer 410. The axis convention assumed is as depicted in Fig. 4.

The third field, *t* indicates the time that the display is to be maintained after rendering of the current layer prior to displaying a next layer. The units for this field are for example, in 1/100ths of a second. Thus a + value of 3 will denote 3/100th of a second. Alternatively, where synchronisation with other media is required they may be assumed to be in units of timer ticks. A timer tick is an arbitrary (or system defined) unit of time. Ticks are often derived from a synchronisation "track" which forms a part of a separate audio or video data file. Therefore a "t" value of 3 in this alternate representation denotes 3 "ticks" of a synchronisation clock.

The fourth field indicates whether or not a corresponding layer is to be removed (and the display restored to its previous state) prior to displaying a next layer. This field is a boolean field and is assigned a "true" or "false"

value depending on whether the output display device 114 is to be or not to be restored to a previous state respectively. If the current layer is to be removed, r = *true*, then the decoder must store the state of the output display device 114 prior to displaying the current layer. On the other hand, if the layer is not to be removed, r = *false* then subsequent layers are composited over the top of (all) previously displayed layers.

The fifth and sixth fields (b and n respectively) shown in table 1, implement a looping mechanism. The fifth (b) field indicates a layer to optionally branch to. This is expressed as a number greater than or equal to zero and is interpreted as the number of layers to go back in the file relative to the next layer. This definition results in a value of $b = 0$ indicating the next layer in the file–the most "natural" progression. A value of $b = 1$ indicates the current layer –a feature which may be used to extend the display time for the corresponding layer beyond what could normally be specified just using the t field. Thus, a "b" value of 3 (b=3) indicates that the sequence of layers to be displayed must loop back two (2) layers before the current layer and redisplay those layers. This is equivalent to repeating the last three (3) layers, the current layer included. If b is 2^{32}, or any value greater than the sequence number of the currently displayed layer (in the file) plus 1, then the layer to branch to is defined to be the first layer in the file.

The sixth field of table 1, n, indicates the number of times the branch indicated by the fifth field, should be followed prior to displaying the next layer in the file sequence. A value of $n = 2^{32}$ is used to indicate that the branch should be followed indefinitely (until explicitly stopped by user or some other higher level control). A person skilled in the art of programming

would realise that 2^{32} is the largest value that n can take due to its representation as a 32 bit integer. Other special values could be used without departing from the spirit of the invention.

Fig. 5 shows a flow chart for the process steps of displaying a multi-layer image according to an embodiment of the present invention. The process starts at block 500. The images file header of a file is read at block 510, which, amongst other information, includes information about the total number of layers. This enables the display area to be determined and initialised at a step 515 for a display device 114 upon which the image data is displayed. A main display loop is initialised at block 520 with the value of a variable denoting a *current layer* being initialised to 1,a value which indicates the first layer in the file. The display loop commences at block 530 with the reading of a corresponding control block for the *current layer* and reading the image data for the current layer. At block 540 the image data for the current layer is displayed as prescribed by information from its corresponding control block. The value recorded as the *current layer* is then updated at block 550. The aim of this step is to establish the index of the layer to be displayed in the next execution of the display loop. Before this can happen the value determined to be the new value for *current layer* is tested at 560 to determine if it is larger than the total number of layers present in the file then execution terminates at block 599. That is, if control block 560 returns true then the display loop exits to block 599. Otherwise control block 560 returns "false" and control returns to block 530 using the newly established value for *current layer*.

Fig. 6 describes in more detail the processing steps executed in block

530 "Read control block for current layer" of Fig. 5. Processing starts at block 800. At decision block 810 a test is performed to determine whether or not this is the first time that this layer's control block has been read (and hence the first time this layer has been displayed. If control block 810 returns "false", indicating that this is the first time this block has been read and that the layer has not yet been displayed, then a variable *c* is instantiated for the layer and set to a value of 0 at block 820 before the actual control parameters for that control block are read in at 830 and processing exits to block 899. This *c* variable is used in later loop calculations hereinafter described with reference to the flow chart shown in Fig. 8. Otherwise decision block 810 returns "true", indicating that the block has been previously read and the layer previously displayed, then previously read parameters for the control block are read at 840 and processing exits to block 899. If block 810 returned true then the value of the variable c is maintained as the same value that entered at start block 800 from a previous loop.

Referring to Fig. 7, there is shown in more detail the process step of the display layer block 540 of Fig. 5. The layer display process starts at step 600. At decision block 610 the *r* parameter (from information field of table 1) from the control block for a current layer is tested to see if its value is "true". If decision block 610 returns "true" then the display area which will be overwritten when the current layer is displayed is saved to an off display storage area at block 620. The layer is then composited over the display area at the point specified by the *x* and *y* parameters (from the layer's control block and as depicted in Fig. 4) at block 630. At block 640 a current state of the

displayed image is then maintained for a period of time specified by the *t* parameter (from the layer's control block). The saved screen area is then restored from the off display storage area at block 650 before exiting at block 699. Otherwise, decision block 610 returns "false", no save and restore operations are required. The layer is composited over the display area at the point specified by the *x* and *y* parameters (from the layer's control block and as depicted in figure 4) at block 630. A current state of the displayed image is then maintained for the period of time specified by the *t* parameter (from the layer's control block) at block 640 before exiting at block 699. The processing then continues at the next execution block, "update value of current layer" 550 of Fig. 5.

The processing involved in the calculation of the new *current layer* variable (block 550 of figure 5) is described with reference to Fig. 8. The process starts at block 700. At decision block 710 the parameter *b* (from the *current layer*'s control block) is tested for a zero value. A zero value indicates that a next layer to be displayed is the next layer in the file. If decision block 710 returns "true" then the value of *current layer* is incremented by one at block 780 and processing exits at block 799. If instead, decision block 710 returns "false" then the value of the current layers variable *c* is tested for a value of "1" at decision block 720. If control block 720 returns true then the value of the current layer's variable is set to zero at block 750, the value of *current layer* is incremented at block 780 and processing exits to block 799. If instead, control block 720 returns "false" then the value of the layers *c* parameter is tested for a value of 2^{32} at control block 730. If control block 730 return "true" then the *current layer* is set to a

value of *current layer* +1 − *b* at block 790 and processing exits to block 799. If instead, control block 730 returns "false" then the value of the layers *c* parameter is tested for a value of 0 at control block 740. If control block 740 returns "true" then the value of *c* is set to be equal to the value of the *current layer*'s *n* parameter (from the layer's control block) in a block 760. Subsequently the *current layer* is set to a value of *current layer* +1 − *b* at block 790 and processing exits to block 799. If instead, control block 740 returns "false" then the value of the *current layer*'s *c* parameter is decremented at block 770. Subsequently the *current layer* is set to a value of *current layer* + 1 − *b* at block 790 and processing exits to block 799.

Second Embodiment

Fig. 9 shows an image file structure in an embodiment of the present invention. The file 1000 comprises a number of elements 1002 − 1008 packed sequentially into a binary file. Elements early in the the file contain header information 1002 which may include information identifying the file type as well as information describing parameters of the image data contained in the file 1000. An element may also describe an extension to the basic file syntax that is not necessarily understood by all file readers.

In the embodiment, each instruction has identical parameterisation and, as a result, has fixed length. This fact can be used by a file reader to determine the instruction boundaries and, where the length of the instruction set is known, the number of instructions. The animation control block 1004 uses the syntax of the file 1000 in which the block is embedded. Usually this provides a mechanism by which the file reader can determine the starting point and length of the control block 1004 as a whole. Each instruction set,

say 1020, (including the leading repeat parameter 1028) is delimited in such a way that the file reader can determine the starting point and length of each set 1020 in a straightforward manner. In the current embodiment, each instruction set is appended to (i) a 32 bit unsigned integer indicating the length of the instruction set and (ii) a 4 byte tag indicating that the ensuing data is a set of animation instructions. This structuring scheme is illustrative, and a different structure, such as a table listing the starting offsets of each instruction set, can equally be used.

The file 1000 contains one or more elements containing image data 1006 or references to image data. There may be several distinct still images 1006 – 1008 contained or referenced in a file and each of these is referred to as a layer. Some of these layers may be visually incomplete when viewed separately as they are intended to be overlayed on or otherwise combined with other image layers in the file for display purposes. Each is however a complete codestream or set of codestreams, capable of being independently decoded and are still considered distinct within the scope of this description. Animation can be performed using one or more of the image layers 1006 – 1008, alone or in combination.

Each image layer eg 1006 comprises one or more channels which may be present as one or more codestreams contained in the file 1000, or referenced by the file or derived by mapping image elements through a lookup table. Each codestream or reference contained in the file 1000 is present in one or more file elements. Information in header elements is used by the file reader to recover the complete codestreams and decode those to image layers.

The channels of each layer comprise arrays of pixel values. These

may correspond to samples of colour information specific to a colour space which is defined within header elements 1002 of the file. A single channel may also correspond to intensity samples as in a greyscale image. One or more channels may also contain samples of opacity information for use in rendering other channels in the layer. This channel is commonly referred to as the alpha channel. Alpha channel data may be binary (or bi-level) with each sample taking on only one of two possible values corresponding to fully transparent and fully opaque. Binary alpha data may be encoded with the colour channels by assigning a unique colour to all pixels which are fully transparent.

This specification discloses a method for describing the animation, comprising a file or codestream 1000 containing a header 1002 with global parameters including but not limited to (i) the screen area (eg 1532 in Fig 10) required to display the animation contained in the file (ii) a block of animation control information 1004 and (iii) a sequence of image layers 1006 – 1008 encoded using any appropriate method.

The animation control information 1004 (also referred to as the animation control block) comprises, as shown in an expanded view 1016, an integer 1014 denoted "tick" defining the duration of a timer tick. The animation control information 1004 also contains an integer 1018 denoted "loop" defining the number of times the animation as a whole should be displayed. The animation control information 1004 further contains one or more sets 1020 – 1022 of frame control instructions. The structure of the animation control block 1004 is described with reference to Table 2.

Field tag	Encoding	Description

Tick	16 bit unsigned integer	The duration in milliseconds of the default timer tick used for interpreting timing instructions. Other temporal measures could be used eg. ticks per second.
Loop	16 bit unsigned integer	The number of times to repeat the display of this animation in its entirety. A value of 2^{16} indicates that the decoder should repeat the animation indefinitely or until stopped by an external signal.
Instruction sets	See Table 2.	Collections of animation instructions

Table 2. Fields contained in the animation control block with descriptions.

A predetermined value of "loop" 1018 can be used to ensure that the animation be repeated an indefinite number of times.

Each of the sets 1020 – 1022 of frame control instructions comprises, as shown in an expanded view 1032, a leading integer 1028 denoted "repeat" indicating the number of times the associated set of instructions 1030 – 1036 should be executed, and a set of instructions which are to be executed by the reader in sequential order. A predetermined value of "repeat" is used to ensure that the animation instruction sequence is executed an indefinite number of times. Table 3 encapsulates the form of the instruction sets 1020 - 1022.

Field tag	Encoding	Description
Repeat	16 bit unsigned integer	The number of times to repeat the execution of the ensuing animation instructions.

Instruction m	See table 3.	Animation instructions

Table 3. Fields contained in each of the "Instruction sets" of the animation control block, with descriptions.

Each instruction say 1034 comprises, as shown in an expanded view 1042 (which comprises two sections 1058 and 1064 in tandem, as depicted by dashed arrows 1060, 1062) an integer 1044 denoted "life" defining the number of timer ticks that should (ideally) occur between completion of execution of the current instruction and completion of execution of the next instruction. The instruction further comprises a binary flag 1046 denoted "persist" defining whether the pixels rendered to screen as a result of execution of the current instruction should appear to persist on the display background or appear to be reset to the pre-execution background. Furthermore, an integer 1048 denoted "next" defines the number of instructions to execute before reusing the current layer where a value of zero implies that the layer shall not be reused for any ensuing instructions notwithstanding execution of a global loop as a result of a non-zero "loop" control.

The first instruction 1030 acts upon the first layer 1006 in the file 1000, and each subsequent instruction acts on the layer specified for that instruction in the "next" field of a previous instruction, or, in the case that no such specification has taken place, the next layer sequentially in the file.

A zero value for "life" (ie 1044) and a false value for "persist" (ie 1046) indicates that the layer being acted upon by that instruction is not rendered in any way by that instruction.

A zero value for "life" (ie 1044) and a true value for "persist" (ie 1046)

indicates that the layer being acted upon by the current instruction is to be considered as part of a frame definition sequence. Such a sequence is terminated upon execution of the next instruction with a non-zero value for "life". Termination of the frame definition sequence results in the composition and display of all of the layers acted upon by the frame definition sequence in such a way that the "persist" and "life" values for the terminating instruction are applied collectively. From a display perspective, all the instructions in a frame definition sequence should appear to be executed as a single instruction.

A predetermined maximum value of "life" (ie 1044) is used to imply that the animation be suspended indefinitely after execution of that instruction. In such cases, execution may be continued as a result of some higher interaction level.

Each instruction (1030) can additionally include an integer pair 1050, 1052 denoted "(x,y)" defining the location to place the top left corner within the display area for the whole image of the layer being acted on by this instruction. The instruction 1030 can also include an integer set 1066 – 1072 denoted "(Cx, Cy, Cw, Ch)" defining the top left corner, width and height of a region to crop from the layer being acted on by this instruction. The cropped region is considered to replace the layer being acted upon within the scope of this instruction only.

Each instruction can additionally include an integer pair 1054, 1056 denoted "(w,h)" defining the width and height of the region within the display area into which the layer being acted upon by this instruction should be rendered. This step includes resampling of the layer if the width and height

of the layer are different to the values specified in the instruction. The form of the instructions 1034 – 1036 is set out in Table 4.

Field tag	Preferred encoding	Description
Persist	1 bit flag	Boolean flag indicating whether the pixels rendered to screen as a result of execution of the current instruction should appear to persist or appear to be reset to the pre-execution background after the instruction's life has expired.
Life	15 bit unsigned integer	The number of timer ticks to aim to place between the completion of this instruction and the completion of the next instruction.
Next	32 bit unsigned integer	The number of instructions to execute (including the current instruction) before reusing the current image layer. A value of zero implies the layer shall not be reused for any ensuing instructions notwithstanding execution of a global loop as a result of a non-zero "loop" control.
x_screen	32 bit unsigned integer	Distance in screen pixels from the left edge of the display area to place the left edge of the layer being acted on by this instruction.
y_screen	32 bit unsigned integer	Distance in screen pixels from the top edge of the display area to place the top edge of the

		layer being acted on by this instruction.
w_screen	32 bit unsigned integer	Width of the display area in screen pixels into which to scale and render the layer being acted on by this instruction.
h_screen	32 bit unsigned integer	Height of the display area in screen pixels into which to scale and render the layer being acted on by this instruction.
x_crop	32 bit unsigned integer	Distance in image pixels to the left edge of a crop region within the layer being acted on by this instruction.
y_crop	32 bit unsigned integer	Distance in image pixels to the top edge of a crop region within the layer being acted on by this instruction.
w_crop	32 bit unsigned integer	Width in image pixels of a crop region within the layer being acted on by this instruction.
h_crop	32 bit unsigned integer	Height in image pixels of a crop region within the layer being acted on by this instruction.

Table 4. Fields contained in the instruction$_m$ fields of the animation control block with descriptions.

The interpretation of the instruction parameters is further explained with reference to Fig. 10.

Fig. 10 shows a virtual screen 1526, upon which a cropped segment 1510 of a layer 1504 is to be rendered, the performance of rendering being depicted by a dashed arrow 1520. The virtual screen 1526 has a width 1524, and a height 1540, these dimensions being referred to an (x,y) origin 1522

depicted by a dark dot. A segment 1532 of the virtual screen 1526, to which a segment 1510 of a layer 1504 is to be rendered, has a width 1530, and a height 1534, denoted respectively by 1054 and 1056 in Fig. 9, these dimensions being referred to an (x,y) origin 1528, denoted 1050, 1052 in Fig. 9. The segment 1510 of the layer 1504 which is to be rendered onto the virtual screen 1526, has a width 1512, and a height 1514, denoted respectively by 1070 and 1072 in Fig. 9, and these dimensions being referred to an (x,y) origin 1508 denoted by 1066, 1068 in Fig. 9. The layer itself 1504 has a width 1506 and a height 1518, these dimensions being referred to an (x,y) origin 1502.

Fig. 11 shows a memory arrangement 1100 for displaying animated image sequences. The memory comprises a viewable memory region 1102 having a capacity equal in size to a screen area defined in the file header 1002, and an off-screen working space 1104, at least equal in size to the largest area that is to be rendered to screen at once (resulting from the execution of a single instruction or the final instruction of a frame definition sequence). The memory further comprises an off-screen backing store 1106, which is again at least equal in size to the largest area that is to be rendered to screen at once (resulting from the execution of a single instruction or the final instruction of a frame definition sequence). The memory further comprises storage 1108 for a list of layers referred to as the "layer memory" which can be used to retrieve a decoded version of any layer explicitly placed in that list. The apparatus further comprises storage 1110 for a list of integers corresponding to the entries in layer memory and containing the number of instructions still to be executed before the corresponding layer (in layer memory) is to be acted upon.

Fig. 12 shows a top level of execution of an embodiment 372 of the animation process. Execution commences at a step 300. A multi-layer file (1000, see Fig. 9) is opened at a step 305, and header and animation control information is read in a step 310, this being described in greater detail with reference to Fig. 13. The header information is used to allocate display resources and support structures in a step 315, this being described in greater detail with reference to Fig. 14. The memory and support structures are initialised in a step 320, this being described in greater detail with reference to Fig. 15. Determination of required instructions is performed in a step 325, which is described in greater detail with reference to Fig. 16. Determination of a required image layer is performed in a step 335, which is described in greater detail with reference to Fig. 17. Rendering a layer in accordance with an instruction is performed in a step 345, which is described in greater detail with reference to Figs 18(a) and 18(b). Flushing of the rendered rectangle to the screen is performed in a step 367, which is described in greater detail with reference to Fig. 19.

The main animation loop of the process 372 begins in a step 325. In this step, the player determines the display instruction to use from the sets of instructions provided in the animation control block. The determined instruction is tested in a step 330 to determine if the instruction has a special value indicating that no further instructions are available. This special value is denoted "stop". If an instruction other than "stop" is found, execution of the process 372 moves to a step 335, where the player determines the image layer which is to be acted upon by the instruction. The determined layer is tested in a step 340, to determine if the determined layer

has a special value, denoted "empty", indicating that no layer could be found.

If there is a layer to use, execution of the process 372 moves to a step 345 where the layer is rendered in accordance with the instruction. In a following step 350, the "next" field of the instruction is tested for a non-zero value which would indicate that the layer is to be used again. If a non-zero value is found then the layer, and the value of the "next" field, are committed to a layer memory at a step 355. The layer memory can take several different, but functionally equivalent forms for the purpose of implementation. In one embodiment, each entry in the layer memory stores the decoded image samples. In another embodiment, the compressed codestream is retained in the layer memory. In a further embodiment, a pointer to the first byte of the layer in the file is stored along with any auxilliary data required in order to read and decode the data. In all cases, the layer memory provides sufficient information to permit the reader to regenerate the pixels of the layer stored therein at some future time.

If the "next" field (ie. 1048) of the current instruction is zero, this implying that the layer is not needed after execution of this instruction, then any memory associated with maintaining that layer's pixel data or decoder state can be freed. In either case, execution of the process 372 subsequently returns to a step 325 where the next instruction is determined, then next layer is determined and rendered and so on.

If at any stage there is no instruction found (ie the step 330 which tests if "instruction is "stop"" returns a "yes" value), or no layer found (ie the step 340 which tests if "layer is "empty"" returns a "yes" value), then the animation sequence 372 is assumed to have finished, and execution moves to

step 360.

If the value of the loop field, tested for at step 360, is zero, then execution of the animation process 372 terminates. However, if the last instruction executed had a zero life field, then there may be undisplayed image data waiting to be rendered in the render image. To logically complete the animation (or alternatively, the still composition) the rendered rectangle of the render image is flushed to screen at a step 367, prior to exiting to the step 370. In an alternate embodiment, the flushing step 367 can be performed prior to the decision step 360. If the loop field is non zero at the step 360, then the loop field value is decremented at a step 365 before re-initialising memory and support structures (at the step 320) and restarting the animation loop.

Step 310 of Fig. 12 is described in detail with reference to Fig. 13. Execution starts at a step 1610. In a subsequent step 1620, a width and height of the screen area used to render the animation is read, along with other header information important to the recovery of image data from the file. Only the width and hight parameters play an integral role in the animation process however. In a following step 1630, top level animation controls are read, these including "tick" and "loop" parameters, as defined in Table 1. Subsequently, in a step 1640, the animation instruction sets are read. In practice, the instruction sets may be read in full, or a pointer into the file maintained for reading during execution of the main animation loop may be read. The header information 1002 as described in relation to Fig. 9 represents only the required minimum set of header information required to implement the embodiment. Other embodiments may incorporate arbitrary

additional header information. Finally, execution exits at a step 1650.

Step 315 of Fig. 12 is desribed in detail with reference to Fig. 14. Execution starts at a step 1710, and proceeds based on information read from the file 1000 (see Fig. 9) at the step 310 of Fig 12. At a step 1720, memory is allocated for a backing store. The purpose of the backing store is to store areas of the screen that must be restored after the display of non persistent frames, i.e. where the persist field of the animation instruction has the value "false". The size of the backing store can be calculated by parsing the instructions, to determine the largest area that will require restoring. Alternatively, the backing store can simply be made the same size as the screen area used to display the animation. No instruction parsing is required in this latter case.

If the animation contains only persistent frames, then no backing store is required, and the step 1720 has no effect. It is noted that information regarding the backing store size can be stored as part of the file header. At a step 1730, memory is allocated for a render image. The purpose of the render image is to act as an off-screen working space in which frames can be composed prior to being copied to screen memory. In the embodiment, the render image is the same size as the screen area used to display the animation. In practice the render image can be smaller, but will usually be at least the same size as the largest screen area updated at any one instant, this being the size of the largest "rendered rectangle" resulting from execution of the instructions. Alternatively, this can be considered in terms of being at least equal in size to the largest area that is to be rendered to screen at once (resulting from the execution of a single instruction or the

final instruction of a frame definition sequence).

It is noted that allocation of a background store is not required if the first layer of the file is fully opaque, and covers the entire image display area (which is specified by the width and height fields of the header). In addition, the backing store is not required if all of the instructions in the animation control block have a persist value of "true". At a step 1740, memory is allocated for the layer memory. Layer memory serves a dual purpose, in that it provides a cache for image data that has already been decoded and rendered but is going to be reused in subsequent frame, and it also provides the mechanism for tracking when the layers contained therein are to be re-used.

To achieve these aims, each entry in layer memory comprises a handle by which the image data for that layer may be retrieved and a variable labelled "next" which records the number of instructions to be executed before reusing the layer.

Finally, at a step 1750, a test is performed to determine if the value of the "loop" field in the animation control block is zero. If this is false (i.e. the step 1750 returns a "no") then the entire animation sequence 372 (see Fig. 12) is to be repeated. In order to support this, an additional background store is allocated at a step 1760 and the initial screen background is copied to this background store in a step 1770. If the value of "loop" is zero (i.e. the step 1750 returns "yes") then there is no need for this background store structure, and execution exits directly to a step 1780.

Step 320 of Fig 12. is described in detail with reference to Fig. 15. Execution starts at a 1800, and at a step 1802, a number of variables are

initialised. Specifically, "restore" is set to "false". This variable indicates when background should be restored from the backing store. The value of "first frame" is set to true, indicating that the first frame of the animation sequence is about to be processed. The "timer" is initialised with the curent time. This variable is used to determine the time at which individual frames of the animation sequence should appear on screen. Finally a variable labelled "rendered rectangle" is initialised to contain four zeros. The rendered rectangle contains the origin (x and y) and size (width and height) of the region in the rendered image that has changed relative to the on-screen display. This is used during screen updates.

At a step 1804, each item in the layer memory is visited, and the items "next" field reset to a value of zero. This value is intended to ensure that the associated image handle will be freed. In a step 1806, a test is performed to determine if a background store is being used, which would imply that the animation sequence 372 (see Fig. 12) is looped at least once. If the step 1806 returns "no", then the screen image can simply be copied to the backing store in a step 1808. If the step 1806 returns a "yes", then the copy of the background contained in the background store must be used, since the screen background may be corrupted with the output from a previous loop execution. This is copied to the backing store at a step 1812. Either way, execution then exits to a step 1810.

Step 325 of Fig. 12 is described in detail with reference to Fig. 16. Execution begins at a step 1900. At a following step 1905, a test is performed to determine if the value of the "first frame" variable is true, which indicates that the animation process 372 (see Fig. 12) is at the very beginning

of the animation sequence. If step 1905 returns "yes", then a "current set" variable is set to point to the first set of instructions defined in the animation control block at a step 1910, and at a following step 1915, the variable "count" is initialised to the value of the repeat field in the aforementioned current set. At a step 1960, the variable "instruction" is set to point to the first instruction in the current set before execution exits at a step 1965.

If the step 1905 returns a "no", indicating that a frame subsequent to the first frame is being animated, then a number of additional tests are required in order to determine which instruction should be used. At a step 1920, a test is performed to determine if "instruction" already points to the last instruction in the current set. If the step 1920 returns a "no", indicating that the end of the current set has not been reached, then "instruction" is incremented, in a step 1940, to point to the next instruction in the current set in sequence order, prior to exiting at the step 1965.

If the step 1920 returns a "yes", indicating that the last instruction in the current set has been performed, then the count variable is tested at a step 1925 to determine if the count variable is zero. If the step 1925 returns a "no", indicating that the instructions in this set should be repeated, then the value of "count" is decremented in a step 1945, "instruction" is set to point to the first instruction in the current set in the step 1960, and execution subsequently exits to the step 1965.

If the step 1925 returns "yes", indicating that any repeats of this instruction set have been completed and that execution should continue with the first instruction of the next instruction set, then a test is performed at a step 1930 to determine if the current set is the last instruction set defined in

the animation control block. If step 1930 returns "yes" – indicating that the current set is the last set – the variable "instruction" is set to a special predetermined value indicating that no further instructions are available. In Fig. 16 this value is denoted "stop". If the step 1930 returns "no", indicating that there are more instruction sets defined by the animation control block still to be processed, then the variable "current set" is set to point to the next instruction set in sequence order at the step 1950, and the variable "count" is initialised to the value of the "repeat" field for that instruction set in a step 1955. Subsequently, the variable "instruction" is set to point to the first instruction in the new current set at the step 1960 before execution exits at the step 1965.

Step 335 of Fig. 12 is described in detail with reference to Fig. 17. Execution starts in a step 2000. In a following step 2005, a variable labelled "current layer" is initialised to a special value denoted "empty" and a variable labelled "N" is initialised to the number of entries in the layer memory. This variable (N) is used in subsequent loop instructions to process each entry in the layer memory. The loop execution starts in a following step 2010, where a test is performed to determine if the value of "N" is zero. If the step 2010 returns "yes", then the loop exits, and the process 335 is directed to a step 2015 where a test is performed to determine if the value of "current layer" has been set to something other than the special value of "empty". If the step 2015 returns "yes", then the pixel data associated with the current layer is retrieved in a subsequent step 2020. In either case, execution susequently exits to a step 2060.

If the step 2010 returns "no", indicating that not all entries in the

layer memory have been visited, then the main body of the loop is executed. At a step 2025, a variable L is set to point to the Nth entry in the layer memory. At a subsequent step 2030, the value of the "next" field in that entry is tested to see if its value is zero. If the step 2030 returns "yes", then the layer is removed from layer memory at step 2035. This is a cleanup step. If the step 2030 returns "no", then a test is performed at a following step 2040 to determine if the value of the "next" field of entry "L" is equal to one and the value of current layer is equal to the special value denoted "empty".

If the step 2040 returns "yes", then the current layer is set to the layer contained in the entry "L", and the "next" field in that entry is set to the value of the "next" field in the current instruction. The value of "N" is subsequently decremented in a step 2055, and execution of the process 335 loops back to the step 2010. If the step 2040 returns "no", then the value of the next field in the entry "L" is decremented, in a step 2050, prior to decrementing "N" at the step 2055 and looping back to the step 2010.

Step 345 of Fig 12 is described in detail with reference to Figs. 18(a) and 18(b). Execution starts at a step 900 in Fig. 18(a). A following step 905 tests for the special case of a non-persistent frame with zero life. If the step 905 returns "yes", indicating that this condition exists, then execution of the process 345 immedately exits to a step 995 (see Fig. 18(b)). If the step 905 returns "no", then the value of the rendered rectangle is updated in a following step 910, to contain the union of the rendered rectangle current value and the screen rectangle defined by the current instruction. At a following step 915, a test is performed to determine if a crop operation is required by the current instruction. If the step 915 returns "yes", indicating

that the crop operation is required, then in one embodiment, the current layer is replaced, only for the scope of the current instruction, with the cropped region at a step 920. In either case, execution then moves to a following step 925.

At the step 925, a test is performed to determine if a rescaling is required by the current instruction. If the step 925 returns "yes", indicating that a rescaling is required, then the current layer is replaced, only for the scope of the current instruction, with a version of the current layer scaled to a width of w_screen and a height of h_screen as defined in the current instruction at step 930. In either case, execution of the process 345 then moves to a step 935, where the current layer is composited over the render image with the top left corner of the current image at the location (x_screen, y_screen) specified by the current instruction. The steps 920, 930, and 935 can be combined in a more optimised fashion in the scope of the present embodiment. In practice, it is likely that an optimised operation that combines one or more of these tasks will be used. The breakdown into individual unambiguous processing steps used in this description is purely for reasons of clarity.

At a following step 940 (see Fig. 18(b)), a test is performed to determine if the layer is persistent and has a life of greater than zero timer ticks. If the step 940 returns a "yes" value, this implies that the render image contains enough information to define the next frame in combination with the current display, and execution moves to a test step 955 where a value of the variable "restore" is tested. If a value of "true" is returned, then the process 345 is directed in accordance with a "yes" arrow to a step 965, in

which the region specified by the rendered rectangle is copied from the render image to the screen image. If the test step 955 returns a "false" value, then the process 345 is directed in accordance with a "no" arrow to a step 960, in which the current screen is copied to the backing store, and the process 345 is thereafter directed to the step 965.

Following the step 965, the process 345 is directed to a step 970, in which the region specified by the rendered rectangle is copied from the backing store to the render image, after which, in a step 975, the rendered rectangle is set to (0,0,0,0), and the variable "restore" is set to "true". The process 345 is then directed to a step 980, which directs the process 345 to wait until the "current time" is greater than a value in the timer.

Returning to the step 940, if a value of "false" is returned from the step, then the process 345 is directed in accordance with a "no" arrow to a step 945, in which the region specified by the rendered rectangle is copied from the render image to the screen image. Thereafter, the value of "restore" is set to "false in a step 950, and the process 345 is directed to the step 980.

After the step 980, the process 345 is directed to a step 985 in which the screen area specified by the rendered rectangle is updated. Thereafter, in a step 990, the timer is set to a value equal to "current time" plus "life", after which the process 345 terminates at the step 995.

Step 367 of Fig 12 is described in detail with reference to Fig. 19. After a commencement step 1300, the process 367 is directed to a step 1310, in which the region specified by the rendered rectangle is copied from the render image to the screen image. Thereafter, in a step 1320, the screen area specified by the rendered rectangle is updated, after which the process

367 terminates in a step 1330.

The method of providing an animation may alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of providing an animation. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories.

Fig. 20 provides an example of an instruction set associated with an animation sequence by which a first image is "slid" onto a second, background image. The figure shows an instruction set 1200 comprising nine columns 1208 – 1224 of integers, and two file designations 1226 and 1228. The uppermost integers 1202 – 1206 of the first three columns 1208 – 1212 provide header information relating respectively to a width and height of the background image, and the number of layers (ie images) used in the animation. The nine columns 1208 – 1224 (with the exception of the first row of 3) refer respectively to the variables x_screen, y_screen, x_crop, y_crop, w_crop, h_crop, "life", "persistence", and "next". Apart from the header information, the columns comprise 11 rows, indicating that the animation is performed in eleven steps. Each row represents an instruction, and the 11 rows represent a single instruction set.

Fig. 21 depicts the animation sequence of Fig. 20. The figure shows the background image 1408, of a disk on a blank background (ie file 1228 in Fig. 20). The dimensions of the image 1408 are 675 (ie 1202 in Fig. 20) wide (depicted by an arrow 1410), and 450 (ie 1204 in Fig. 20) high (depicted by an arrow 1406). The figure also shows an image 1404 (ie 1226 in Fig. 20) which is to be slid onto the background image 1408. The image 1404 has a width

and height depicted by arrows 1400 and 1402 respectively. Four further views of the background image are provided, with the image 1404 slid successively further onto the background image.

A first row 1230 of the instruction set 1200 (see Fig. 20) lays down the background image 1412. The first two integers from the left of the row 1230 which are the x_screen and y_screen values, indicate that the image is to be positioned with it's top left corner at the top left corner of the display area. Since the "next" value of this row 1230, ie the right-most integer, has a value of "0", this indicates that this present image, or layer, will not be re-used, and that the subsequent image, in this case the image 1404, is the next one to process.

The next row 1232, consequently processes the image 1404. The first two integers from the left of the row 1232 which are the x_screen and y_screen values, indicate that the image 1404 is to be positioned with it's top left corner at the top left corner of the display area. The third and fifth integers from the left of the row, ie x_crop and w_crop indicate that part of the image 1404 which is to be "preserved" in the x direction for the present instruction. This is performed by moving x_crop (ie 400) along the image 1404 from the left, and preserving the next w_crop (ie 45) of the image. Similarly, the fourth and sixth integers from the left of the row, ie y_crop and h_crop indicate that part of the image 1404 which is to be "preserved" in the y direction for the present instruction. This is performed by moving y_crop (ie 000) down the image 1404 from the top, and preserving the next h_crop, ie 124, which is, in the present case, the entire image. Accordingly, the rightmost "45", running the full height "124" of the image, is to be preserved,

and this is positioned at x_screen, y_screen ie at a (0,0) displacement from the top left hand origin. The result of this is depicted by 1414 in the figure 20, which shows the image 1404 partially slid onto the background image. Still considering the row 1232 the seventh integer from the left, ie "life", has a value of 1, indicating that a single tick should occur between completion of execution of the present instruction, and completion of execution of the next instruction. This value of "life" results in a uniform sliding movement of the image 1404.

Still considering the row 1232 the eighth integer from the left, ie "persist", has a value of 0, meaning that the screen value is reset to the pre-execution background prior to the execution of the next instruction.

The right-most column of the instruction row 1232 gives the value of "next" to be 1, meaning that the current layer (image 1404) is to be used with the next instruction, where a slightly longer area is cropped and rendered over the top left corner of the background.

Progressive stages of the image 1404 being slid onto the image 1408, are shown in 1416 and 1418.

Industrial Applicability

It is apparent from the above that embodiments of the invention are applicable to the computer and data processing industries, and in particular to segments of these industries. Furthermore, embodiments of the invention are also applicable to the advertising and entertainment industries.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being

illustrative and not restrictive.

4. Brief Description of the Drawings

A number of preferred embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1 is a schematic block diagram of a general-purpose computer with which embodiments of the present invention can be practiced;

Fig. 2 is a diagram illustrating an example of a file format according to an embodiment of the present invention;

Fig. 3 is a diagram illustrating another example of a file format according to an embodiment of the present invention;

Fig. 4 is a diagram illustrating the axes arrangement used in an embodiment of the present invention;

Fig. 5 is a flow chart of the overview of the flow control for a looping mechanism of an embodiment;

Fig. 6 is a flow chart of the "read control block for current layer" step of Fig. 5 in more detail;

Fig. 7 is a flow chart of the "display layer" step of Fig. 5 in more detail; and

Fig. 8 is a flow chart of the "Update value of current layer" step of Fig. 5 in more detail.

Fig. 9 shows an image file structure in accordance with an embodiment of the invention;

Fig. 10 shows a virtual screen, upon which a segment of a layer is to be rendered;

Fig. 11 shows an memory arrangement to support displaying animated image sequences in accordance with an embodiment of the invention;

Fig. 12 is a flow diagram of method steps, showing an animation process in accordance with an embodiment of the invention;

Fig. 13 is a flow diagram of method steps relating to the step of reading header and animation control block information in Fig. 12;

Fig. 14 is a flow diagram of method steps relating to the step of allocation of screen memory and support structures in Fig. 12;

Fig. 15 is a flow diagram of method steps relating to the step of initialisation of memory and support structures in Fig.12;

Fig. 16 is a flow diagram of method steps relating to the step of instruction determination in Fig. 12;

Fig. 17 is a flow diagram of method steps relating to the step of determination of image layer in Fig. 12;

Fig. 18 is a flow diagram of method steps relating to the step of rendering of image layers in Fig. 12;

Fig. 19 is a flow diagram of method steps relating to the step of flushing of a rendered rectangle to the screen in Fig. 12.

Fig. 20 shows an example of an image file structure in accordance with an embodiment, associated with an animation sequence; and

Fig. 21 depicts the animation sequence of Fig. 20.

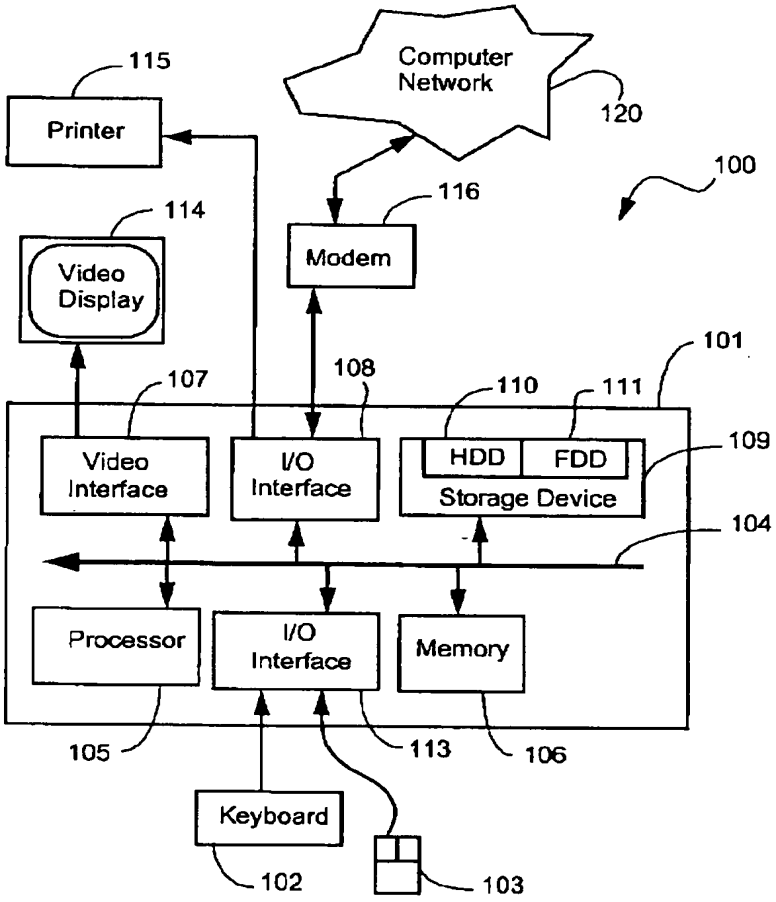

Computer Network
120
115
Printer
100
116
Modem
114
Video Display
101
108
107
110
111
109
Video Interface
I/O Interface
HDD
FDD
Storage Device
104
Processor
I/O Interface
Memory
105
113
106
Keyboard
102
103

Fig. 1

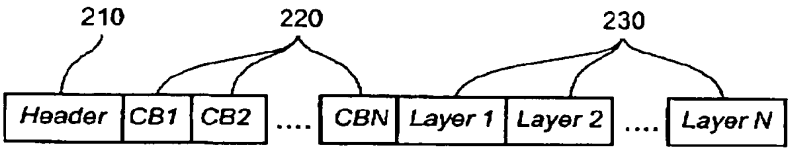


Fig. 2

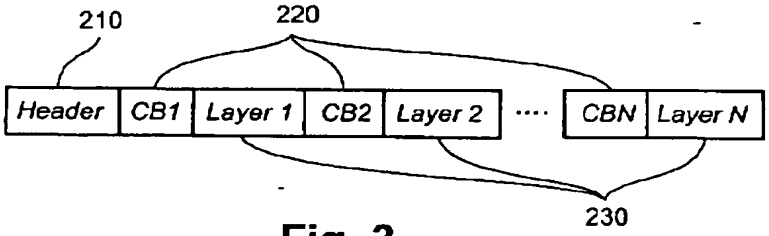


Fig. 3

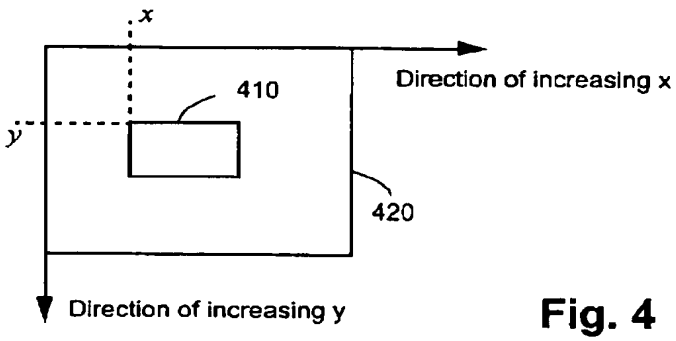


Fig. 4

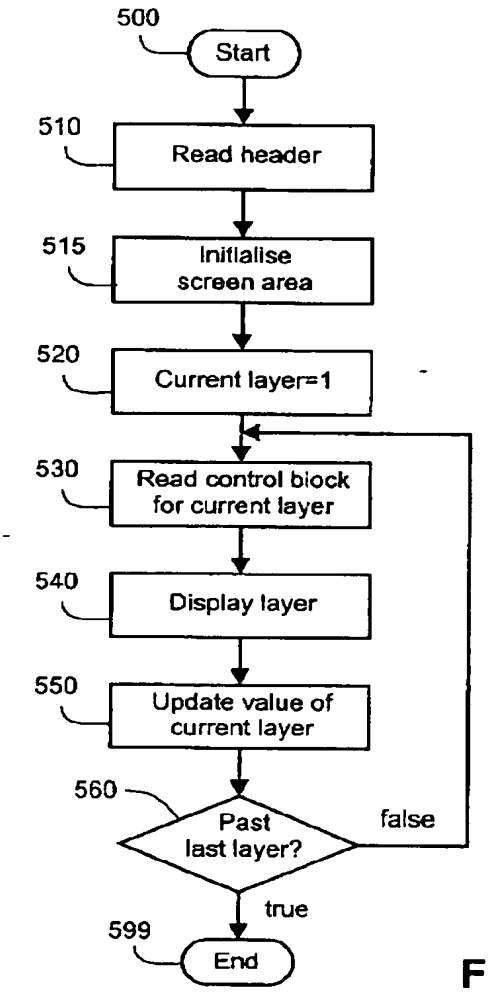

500
Start
510
Read header
515
Initialise screen area
520
Current layer=1
530
Read control block for current layer
540
Display layer
550
Update value of current layer
560
Past last layer?
false
true
599
End

Fig. 5

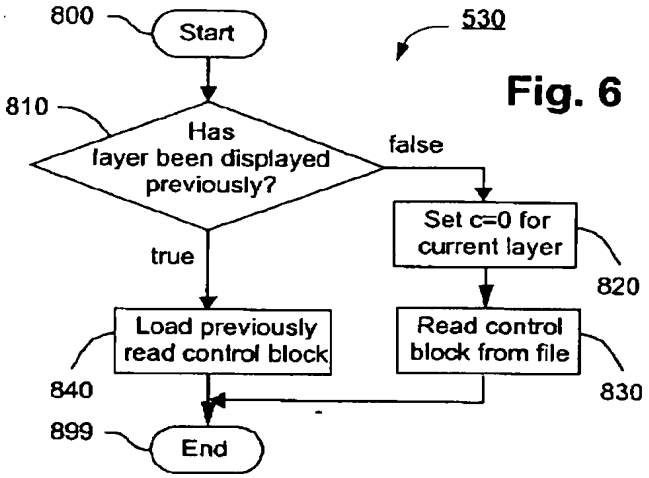

800
Start
530
810
Has layer been displayed previously?
false
true
Set c=0 for current layer
820
Load previously read control block
Read control block from file
840
830
899
End

Fig. 6

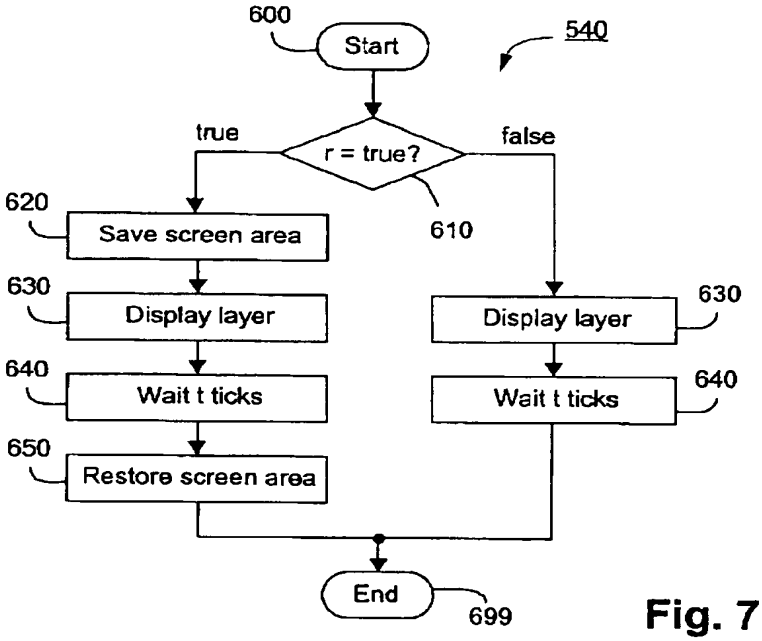

600
Start
540
true
r = true?
false
610
620
Save screen area
630
Display layer
630
640
Wait t ticks
640
650
Restore screen area
End
699

Fig. 7

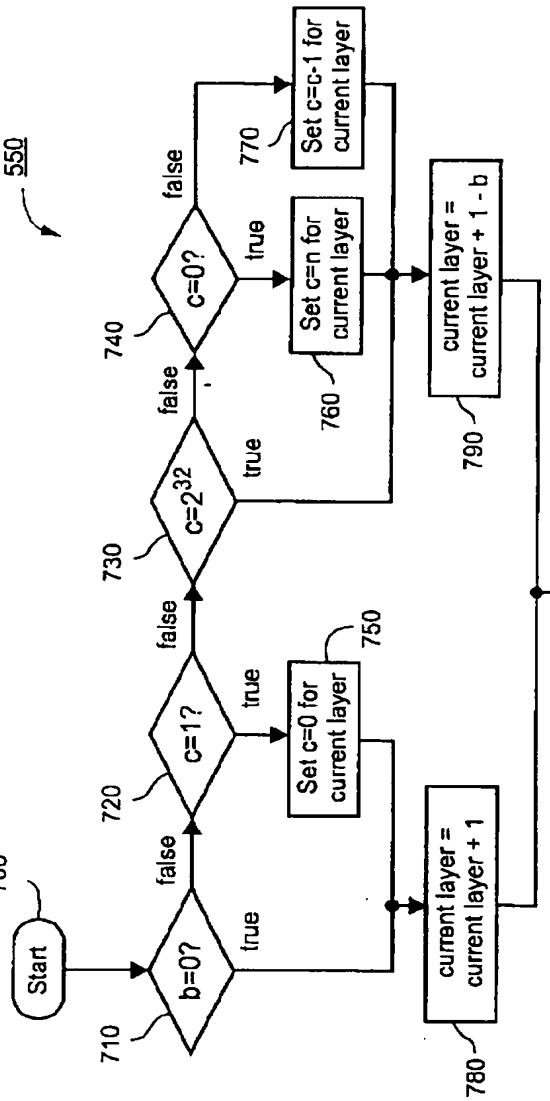
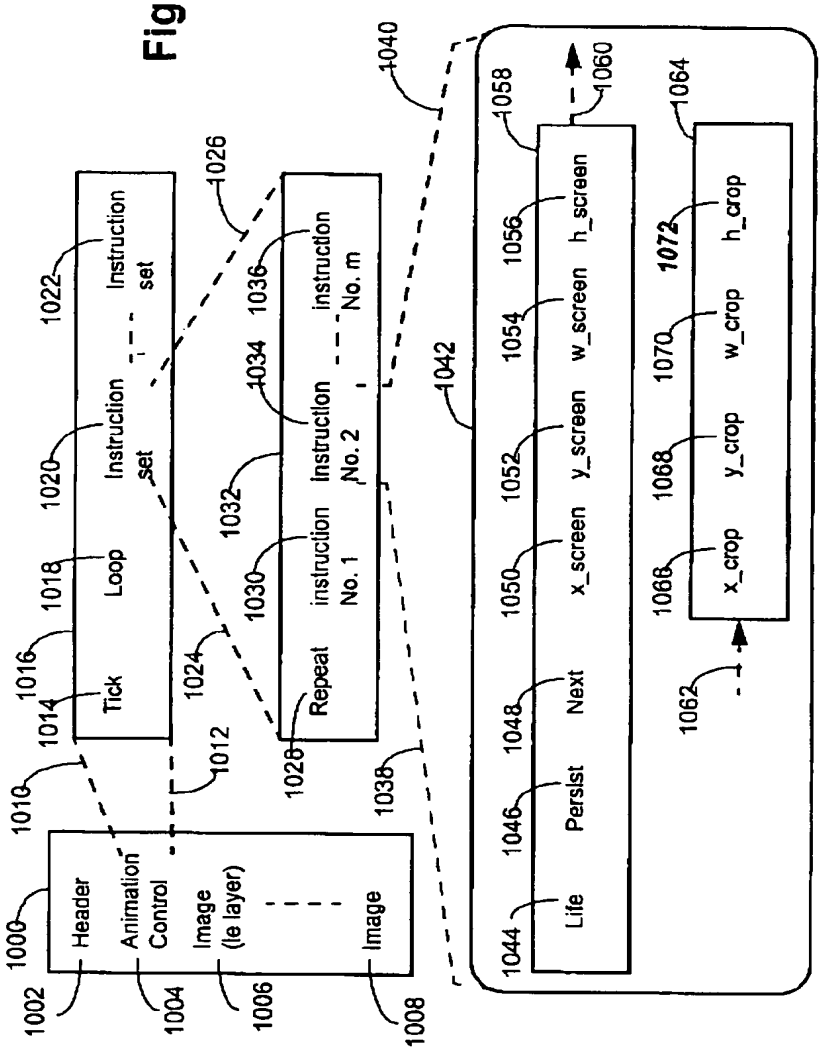

550
700
Start
710
b=0?
false
true
720
c=1?
false
true
730
c=2^32
false
true
740
c=0?
false
true
750
Set c=0 for current layer
760
Set c=n for current layer
770
Set c=c-1 for current layer
780
current layer = current layer + 1
790
current layer = current layer + 1 - b
799
End

Fig. 8

Fig. 9


1000
Header
1002
1004
Animation Control
1006
Image (Ie layer)
1008
Image
1010
1012
1014
Tick
1016
1018
Loop
1020
Instruction set
1022
Instruction set
1024
1026
1028
Repeat
1030
instruction No. 1
1032
Instruction No. 2
1034
1036
instruction No. m
1038
1040
1042
1044
Life
1046
Persist
1048
Next
1050
x_screen
1052
y_screen
1054
w_screen
1056
h_screen
1058
1060
1062
1064
1066
x_crop
1068
y_crop
1070
w_crop
1072
h_crop

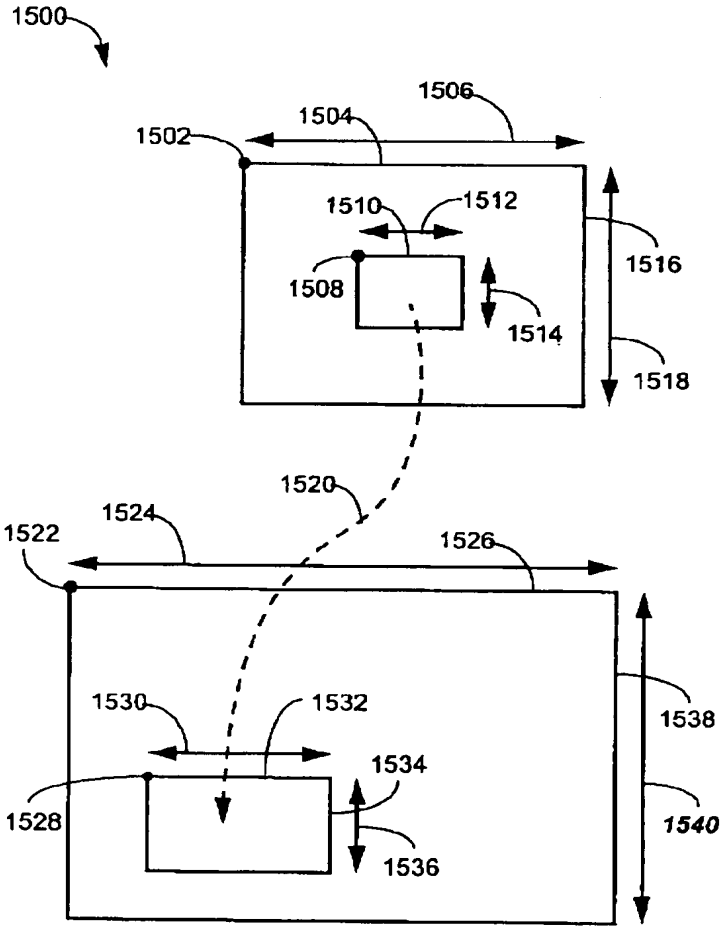

1500
1502
1504
1506
1508
1510
1512
1514
1516
1518
1520
1522
1524
1526
1528
1530
1532
1534
1536
1538
1540

Fig. 10

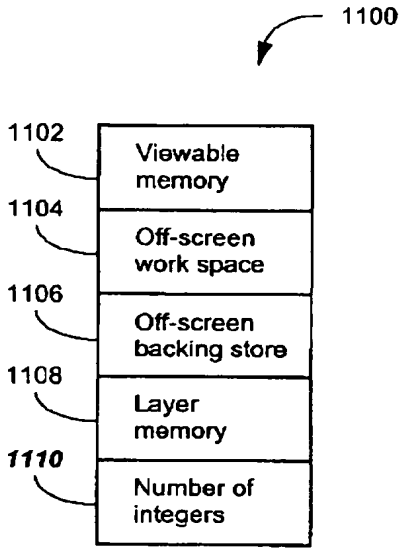

1100
1102
Viewable memory
1104
Off-screen work space
1106
Off-screen backing store
1108
Layer memory
1110
Number of integers

Fig. 11

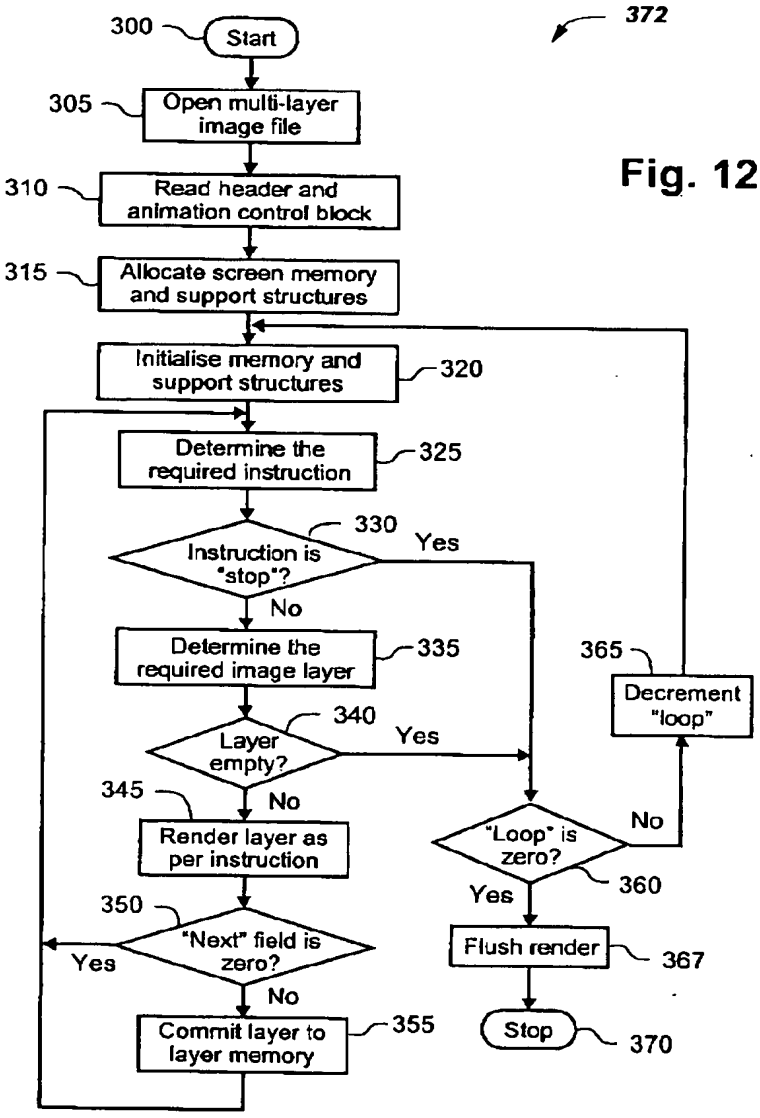

372
300
Start
305
Open multi-layer image file
310
Read header and animation control block
315
Allocate screen memory and support structures
Initialise memory and support structures
320
Determine the required instruction
325
330
Instruction is "stop"?
Yes
No
Determine the required image layer
335
340
Layer empty?
Yes
345
No
Render layer as per instruction
350
"Next" field is zero?
Yes
No
Commit layer to layer memory
355
365
Decrement "loop"
"Loop" is zero?
No
360
Yes
Flush render
367
Stop
370

Fig. 12

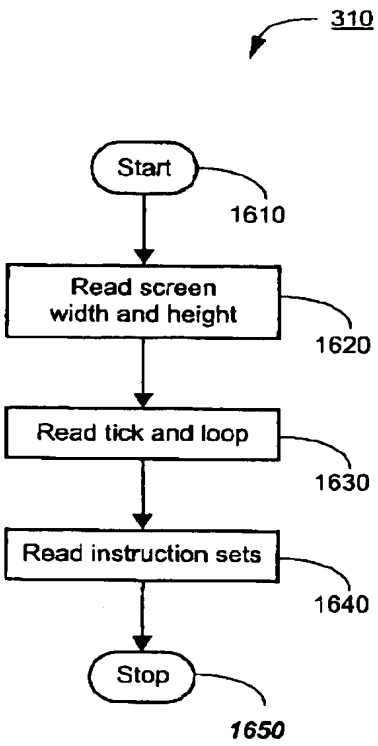

310
Start
1610
Read screen width and height
1620
Read tick and loop
1630
Read instruction sets
1640
Stop
1650

Fig. 13

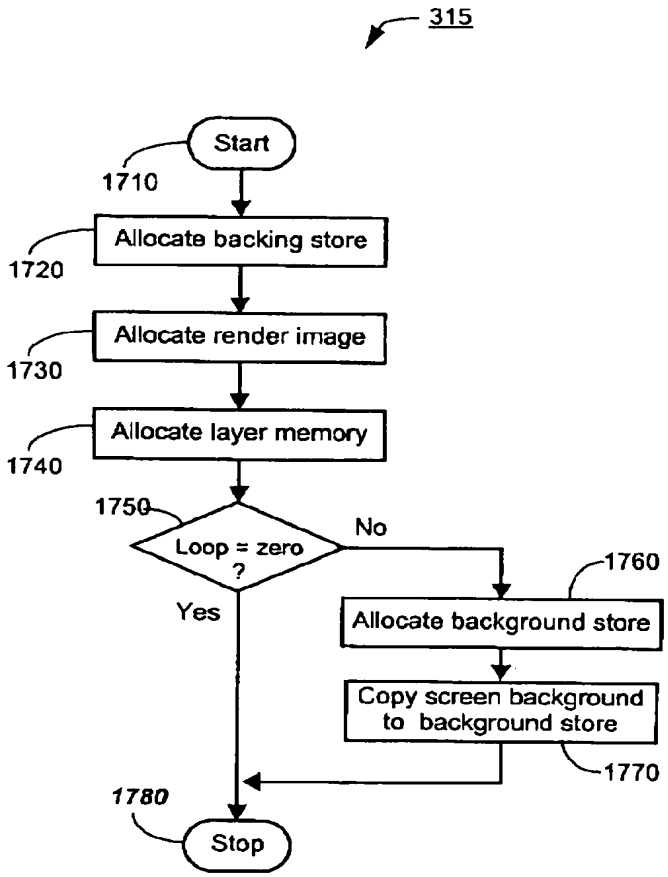

315
Start
1710
Allocate backing store
1720
Allocate render image
1730
Allocate layer memory
1740
1750
Loop = zero ?
No
Yes
1760
Allocate background store
Copy screen background to background store
1770
1780
Stop

Fig. 14

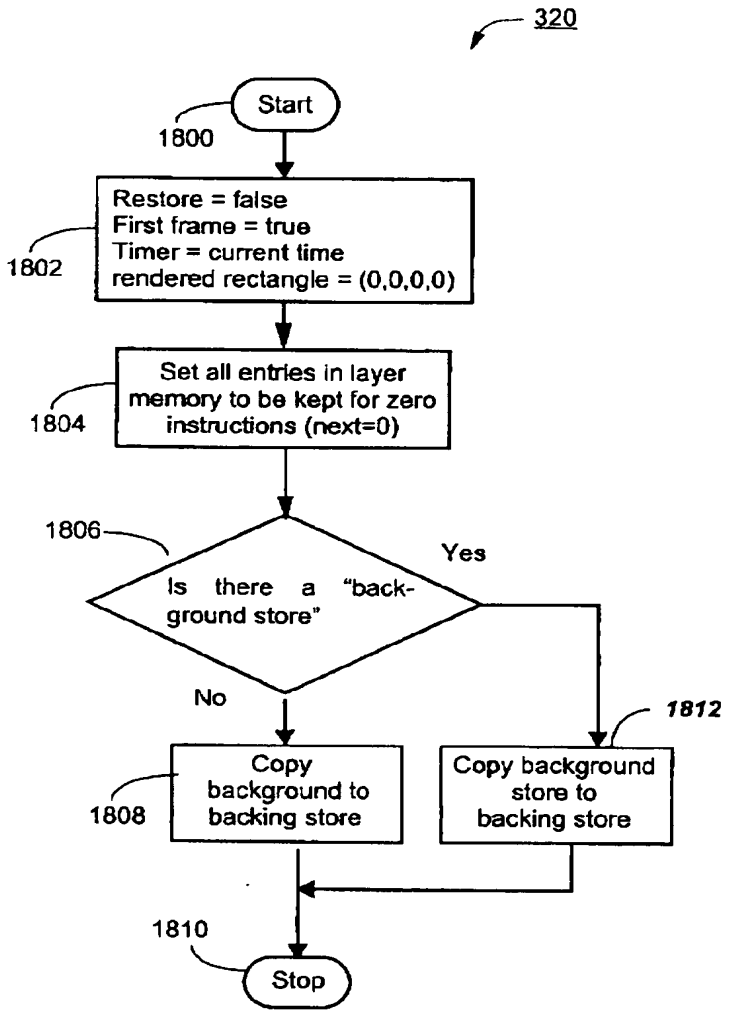

320
Start
1800
Restore = false
First frame = true
Timer = current time
rendered rectangle = (0,0,0,0)
1802
Set all entries in layer memory to be kept for zero instructions (next=0)
1804
1806
Is there a "background store"
Yes
No
1812
Copy background to backing store
1808
Copy background store to backing store
1810
Stop

Fig. 15

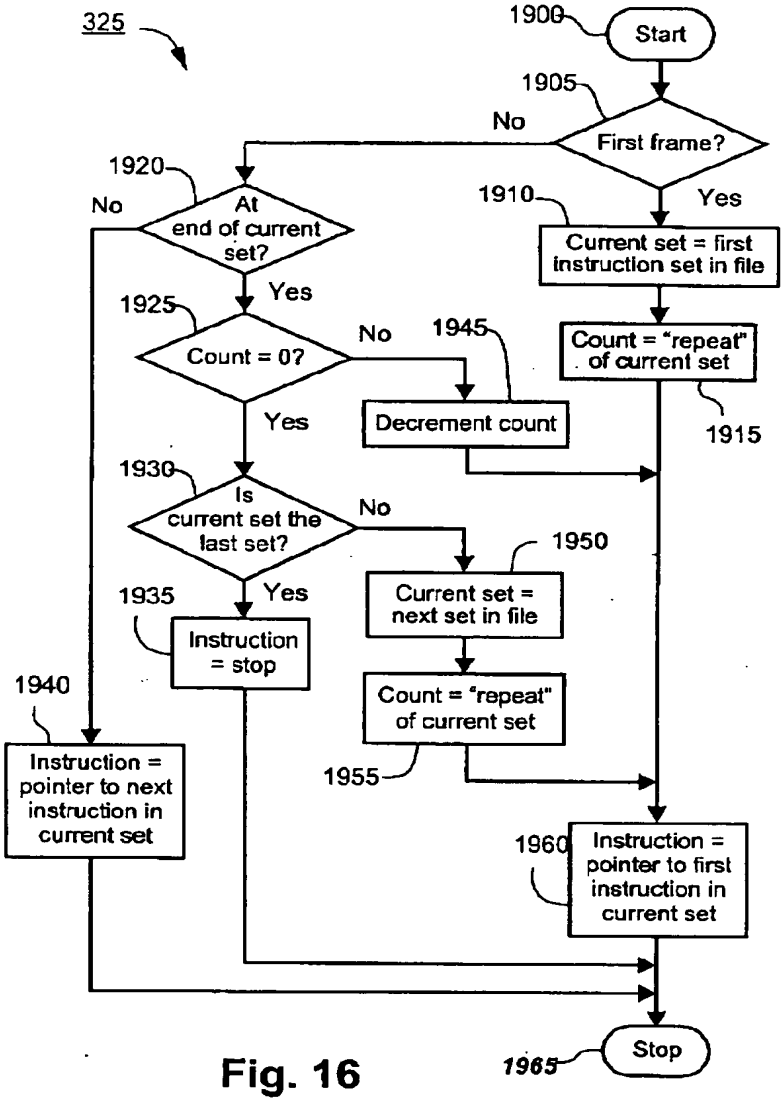
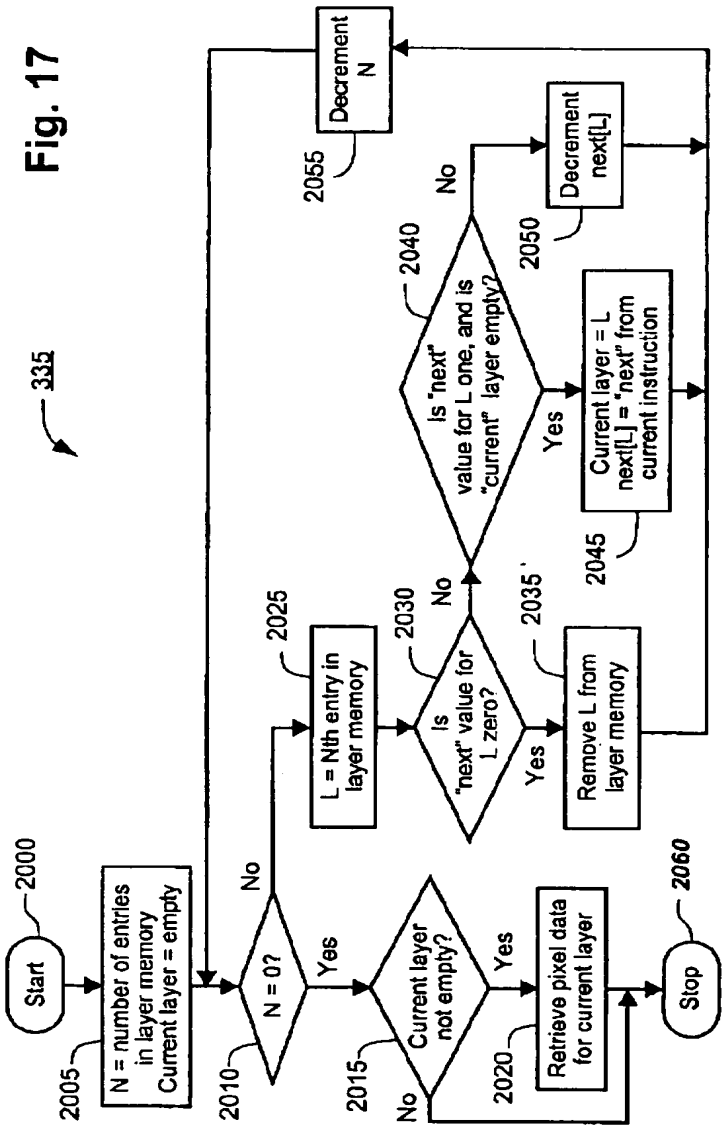

325
1900
Start
1905
First frame?
No
Yes
1910
Current set = first instruction set in file
1920
At end of current set?
No
Yes
1925
Count = 0?
No
1945
Decrement count
Yes
Count = "repeat" of current set
1915
1930
Is current set the last set?
No
1950
Current set = next set in file
Yes
1935
Instruction = stop
Count = "repeat" of current set
1940
Instruction = pointer to next instruction in current set
1955
1960
Instruction = pointer to first instruction in current set
Stop
1965

Fig. 16

Fig. 17


335
Start
2000
N = number of entries in layer memory Current layer = empty
2005
N = 0?
2010
No
Yes
L = Nth entry in layer memory
2025
Is "next" value for L zero?
2030
No
Yes
Remove L from layer memory
2035
Current layer not empty?
2015
No
Yes
Retrieve pixel data for current layer
2020
Is "next" value for L one, and is "current" layer empty?
2040
No
Yes
Decrement next[L]
2050
Current layer = L next[L] = "next" from current instruction
2045
Decrement N
2055
Stop
2060

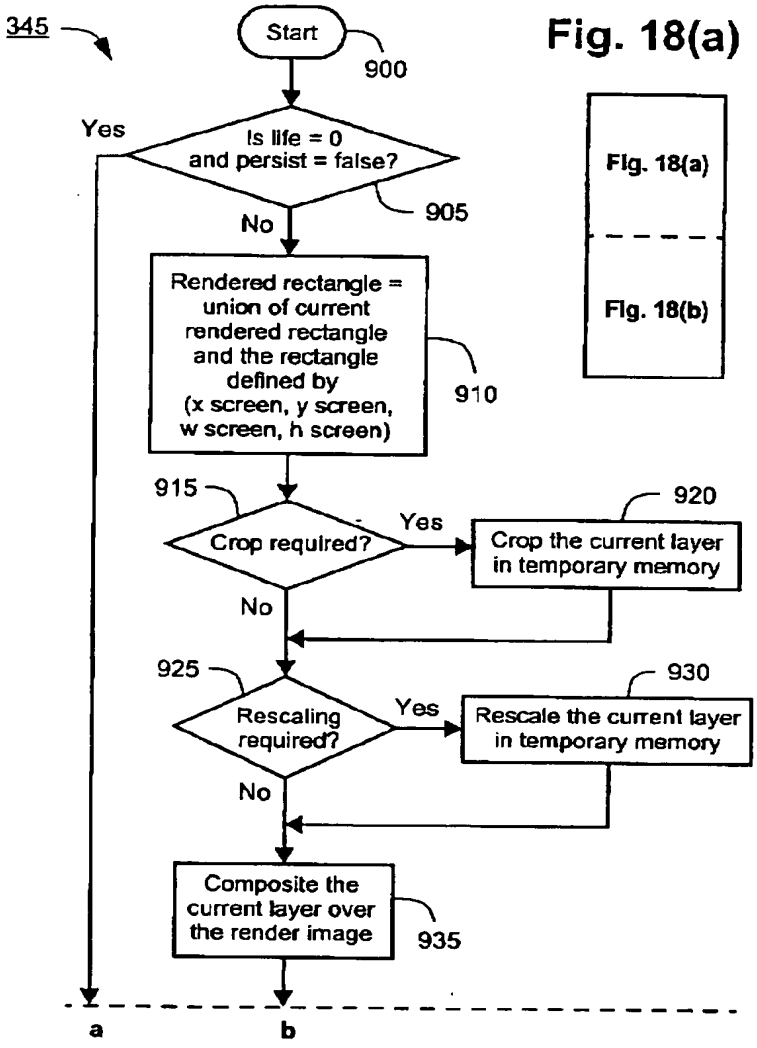

345
Start
900
Yes
Is life = 0
and persist = false?
905
No
Rendered rectangle =
union of current
rendered rectangle
and the rectangle
defined by
(x screen, y screen,
w screen, h screen)
910
915
Crop required?
Yes
920
Crop the current layer
in temporary memory
No
925
Rescaling
required?
Yes
930
Rescale the current layer
in temporary memory
No
Composite the
current layer over
the render image
935
a
b

Fig. 18(a)

Fig. 18(a)
Fig. 18(b)

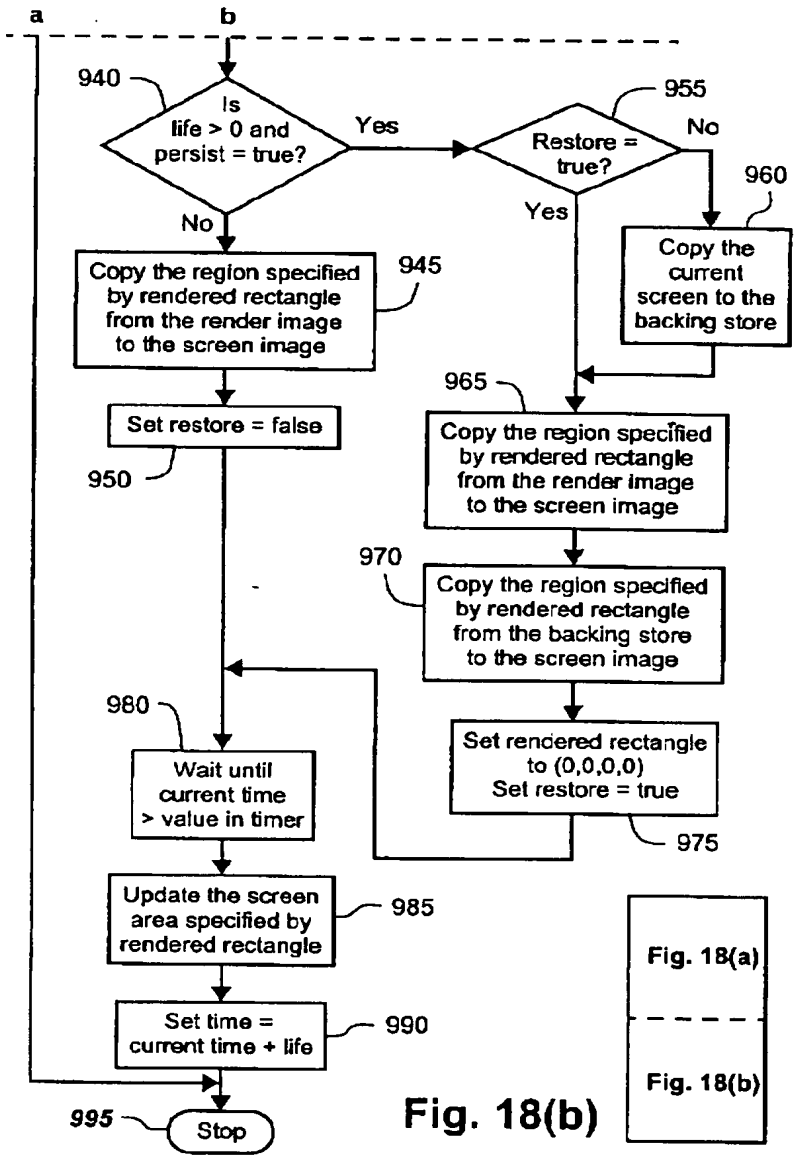

a
b
940
Is life > 0 and persist = true?
Yes
No
955
Restore = true?
No
Yes
960
Copy the current screen to the backing store
945
Copy the region specified by rendered rectangle from the render image to the screen image
965
Copy the region specified by rendered rectangle from the render image to the screen image
Set restore = false
950
970
Copy the region specified by rendered rectangle from the backing store to the screen image
980
Wait until current time > value in timer
Set rendered rectangle to (0,0,0,0) Set restore = true
975
Update the screen area specified by rendered rectangle
985
Set time = current time + life
990
995
Stop
Fig. 18(a)
Fig. 18(b)

Fig. 18(b)

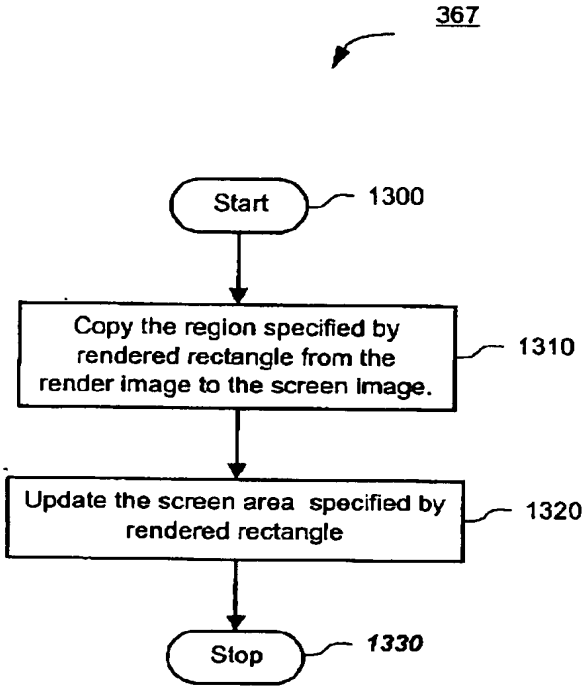

367
Start
1300
Copy the region specified by rendered rectangle from the render image to the screen image.
1310
Update the screen area specified by rendered rectangle
1320
Stop
1330

Fig. 19

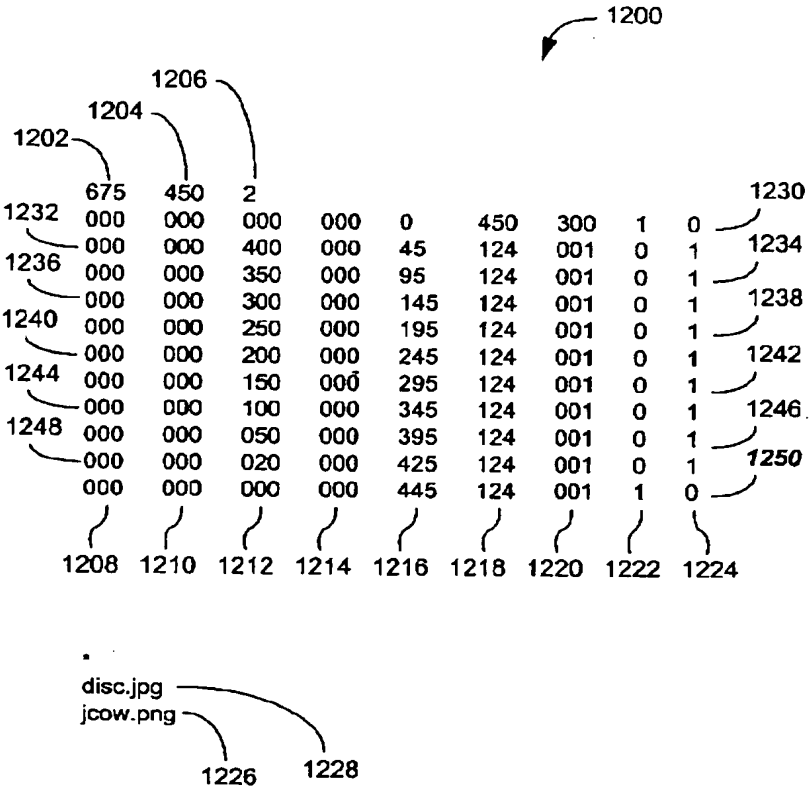
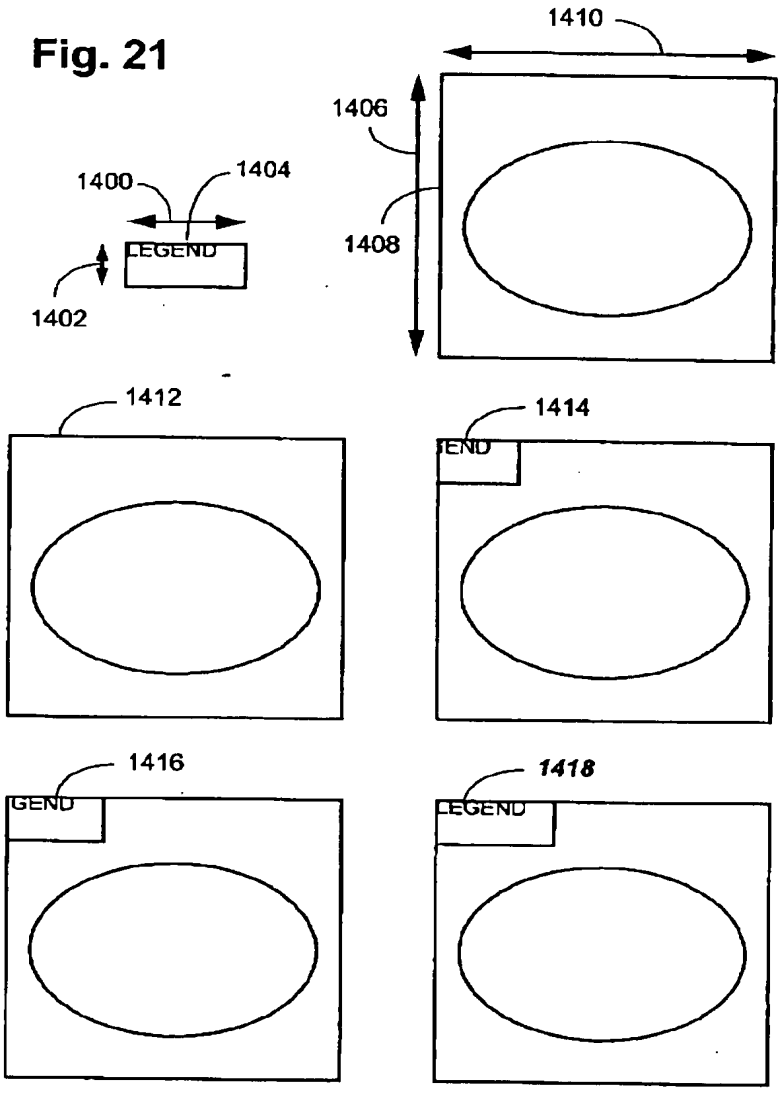

1200
1206
1204
1202
675 450 2
1230
1232
000 000 000 000 0 450 300 1 0
000 000 400 000 45 124 001 0 1
1234
1236
000 000 350 000 95 124 001 0 1
000 000 300 000 145 124 001 0 1
1238
1240
000 000 250 000 195 124 001 0 1
000 000 200 000 245 124 001 0 1
1242
1244
000 000 150 000 295 124 001 0 1
000 000 100 000 345 124 001 0 1
1246
1248
000 000 050 000 395 124 001 0 1
000 000 020 000 425 124 001 0 1
1250
000 000 000 000 445 124 001 1 0
1208 1210 1212 1214 1216 1218 1220 1222 1224
disc.jpg
jcow.png
1226
1228

Fig. 20

Fig. 21



Abstract

A method of processing a multi-layer image file (100) comprising a first plurality of image layers (106-108), and a second plurality of control blocks (120-122) is disclosed. The processing produces an animation sequence. The method comprises processing an image layer (eg 106) in accordance with a corresponding control block (eg 120), thereby providing an image for said animation sequence. The method further comprises tagging the image layer (106) for reprocessing, if the image layer (106) is to be used again in the image sequence, said tagging using a relative address referred to one of (i) an address of said corresponding control block (120) and (ii) an address of a control block corresponding to another image layer.